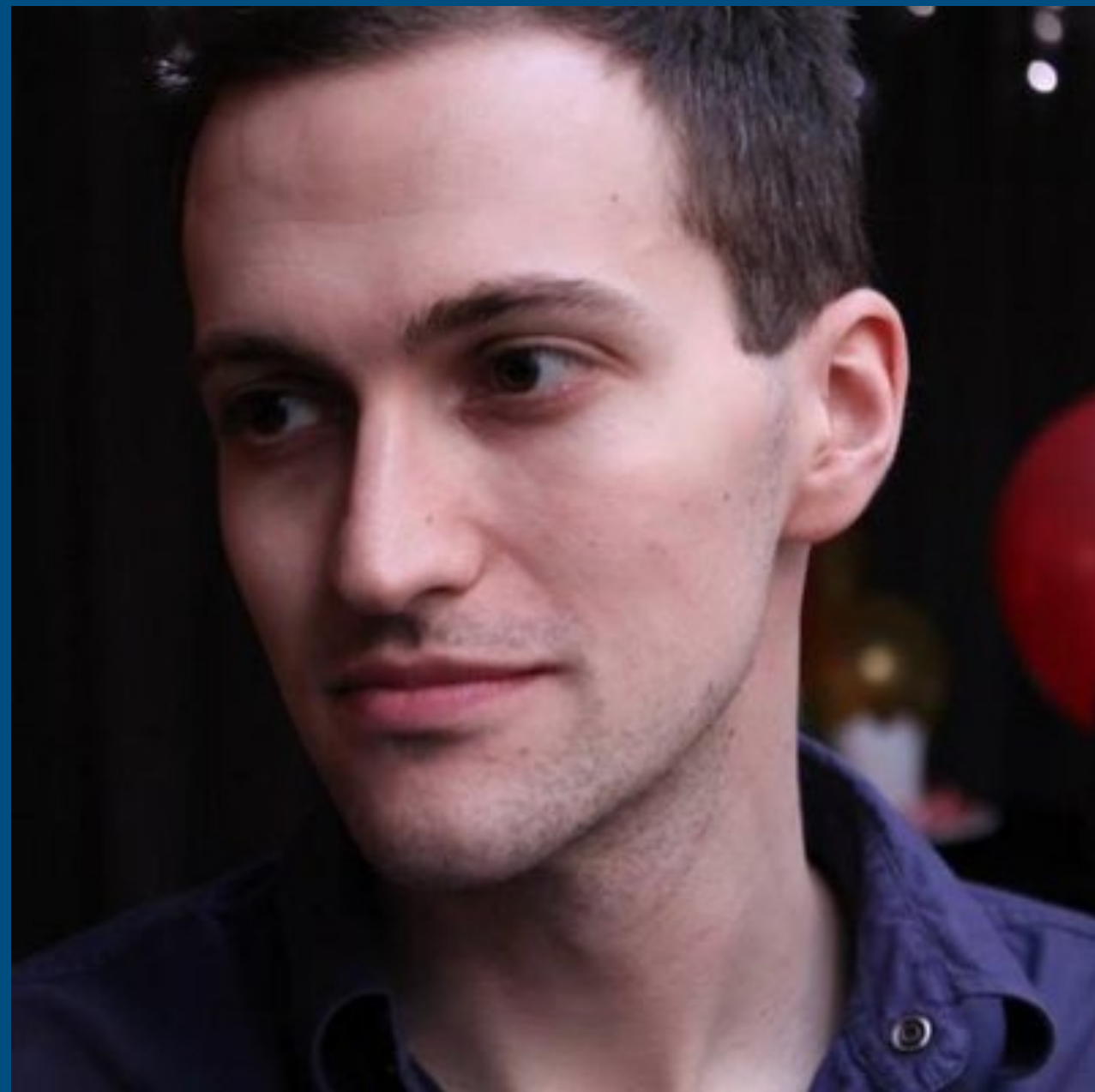


# Dancing with Serverless

Peter Sbarski | @sbarski



# About Me



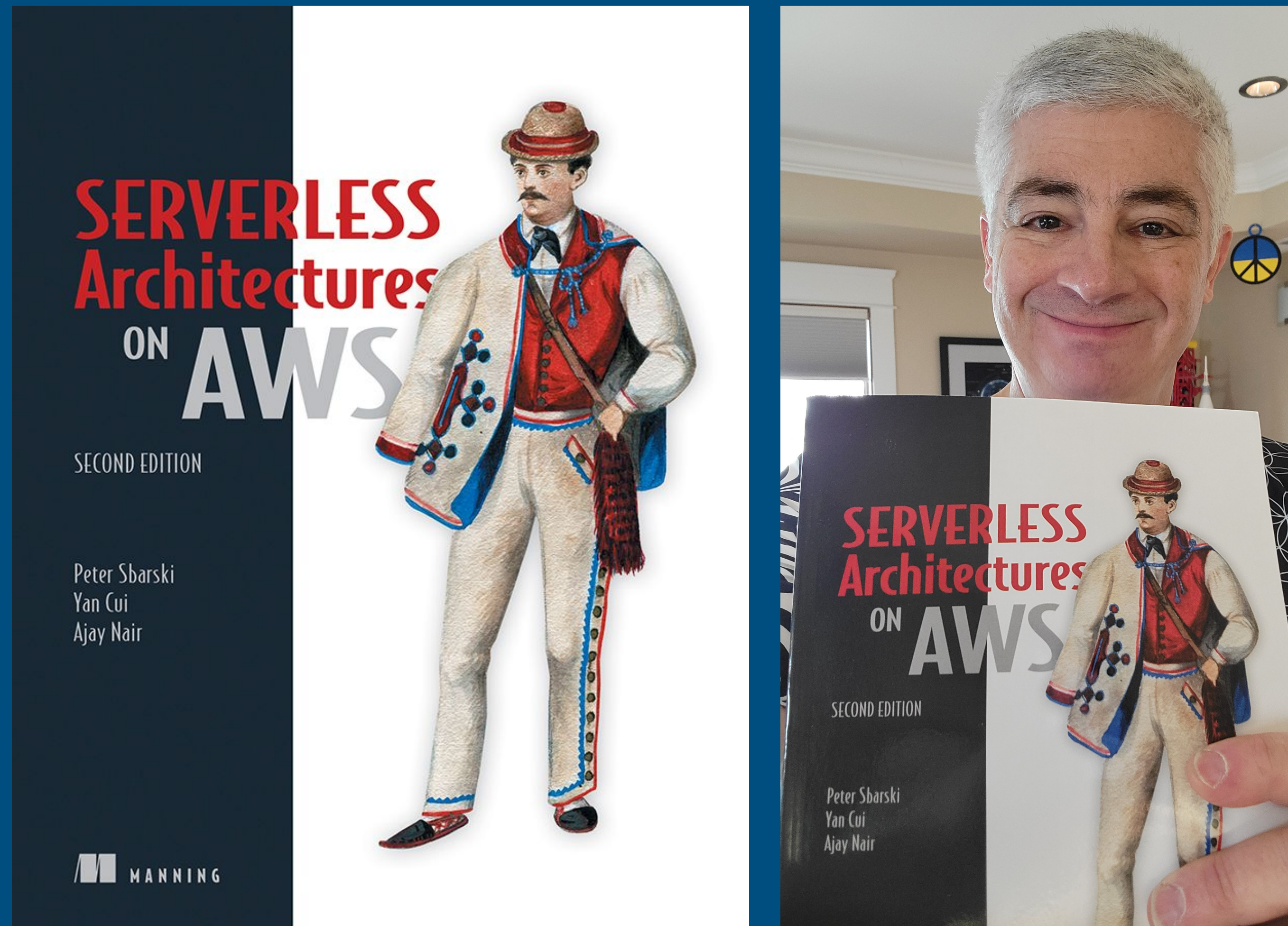
- AWS Serverless Hero
- Author of Serverless Architectures on AWS
- Organiser of the Melbourne Serverless Meetup
- Former VP Education & Research at A Cloud Guru
- Former head of Serverlessconf



- Building something new... reveal later



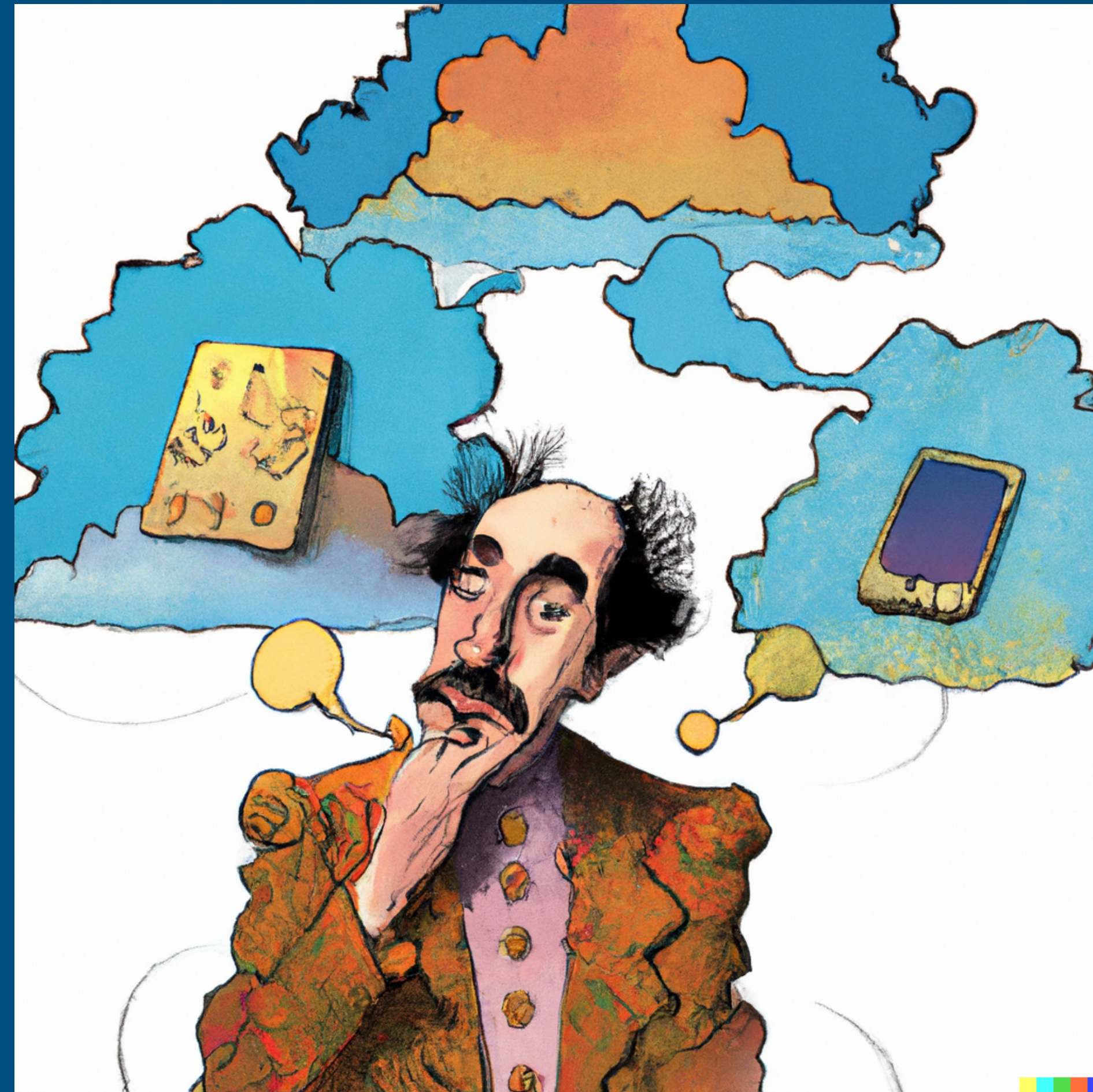
# Serverless Architectures on AWS (2nd Edition)





# I am thinking about this talk

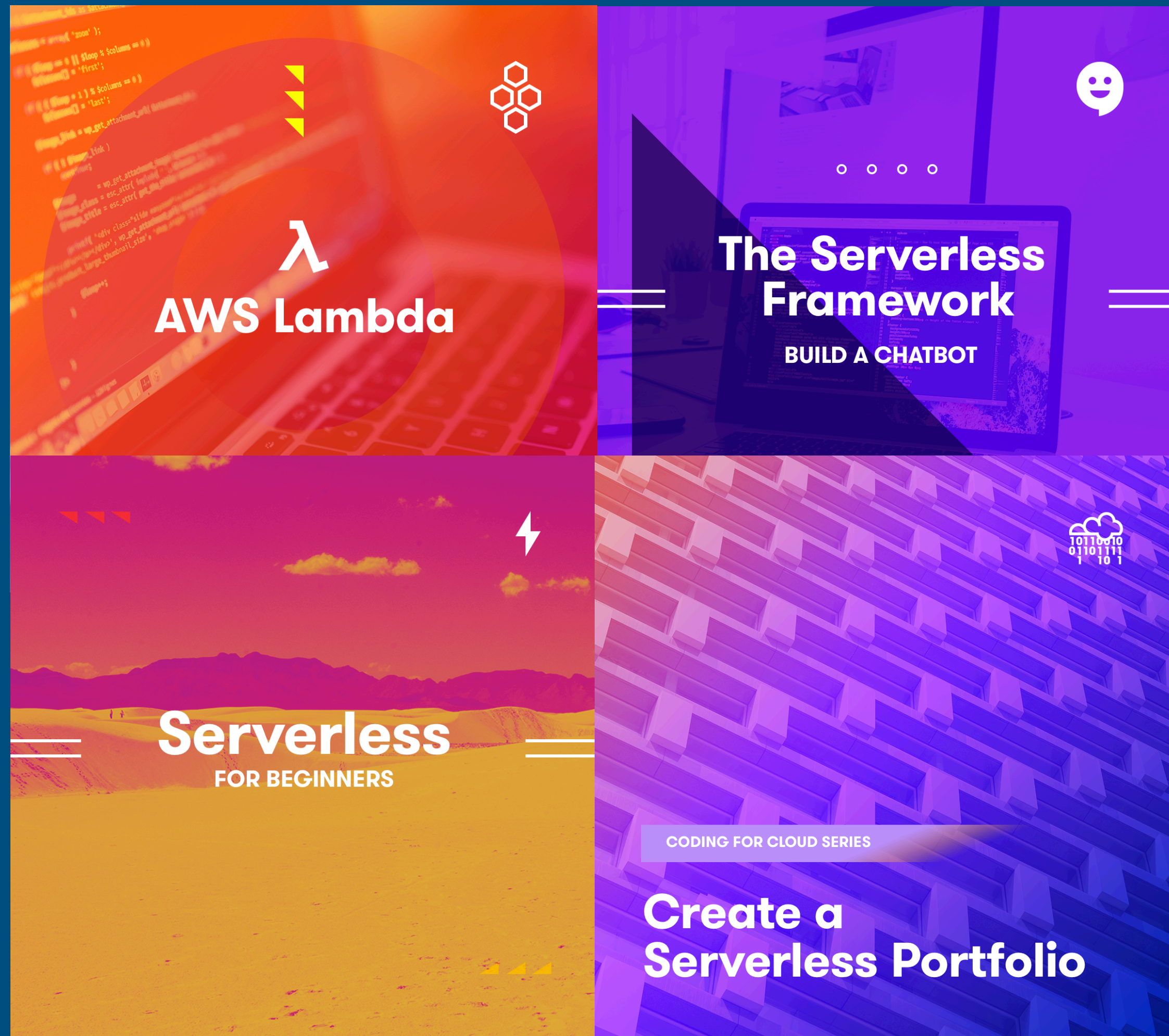
According to Dall E





# Early Days at ACG

## Teaching the world to cloud



Video Lessons  
Quiz Engine  
Online Store  
Sign Up / Login  
Scale Effortlessly  
Low Operational Overhead



# Starting up in 2015

## Serverless Monolith

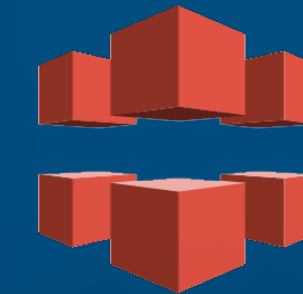


Auth0

Authentication with Auth0



Website (SPA)



Cloudfront



Videos in S3



API Gateway

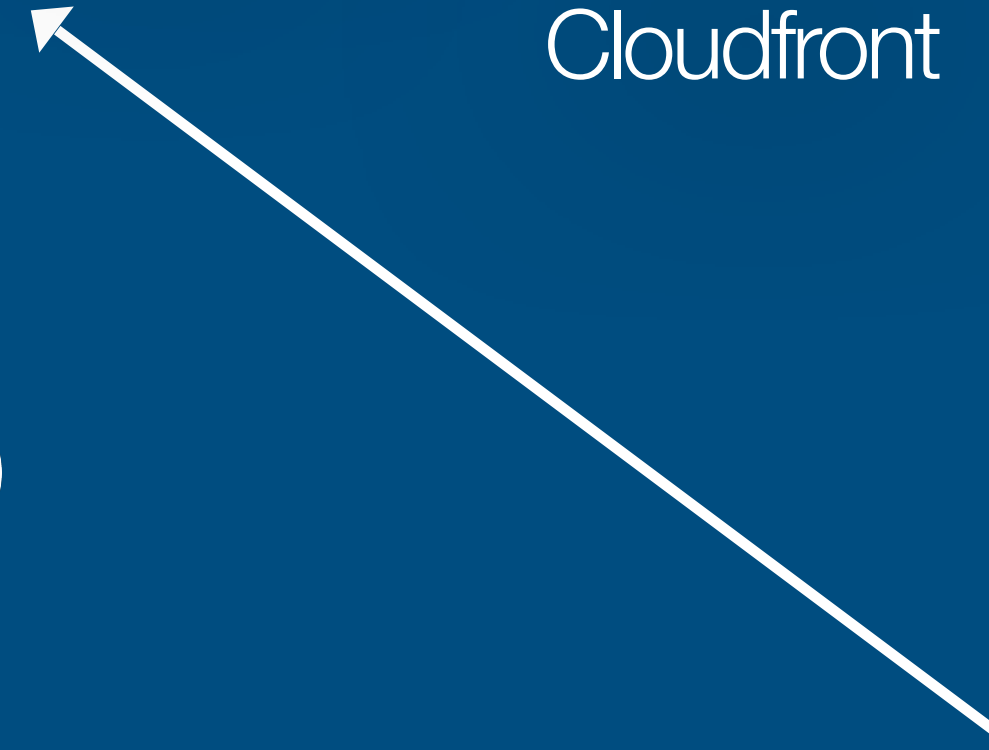


Lambda



Firebase

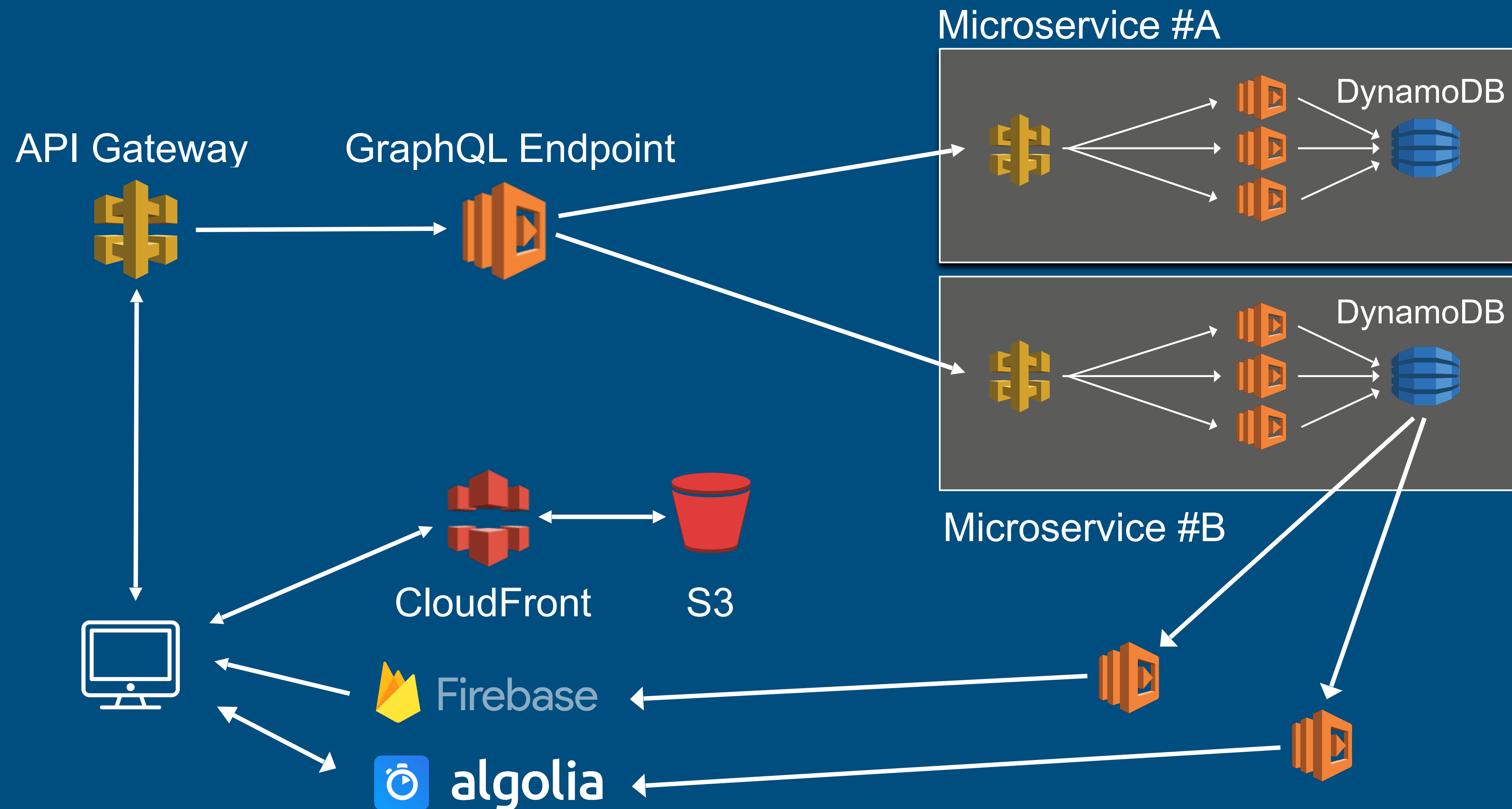
Our database





# Evolution

Teams of developers working in parallel





# We evolved our architecture

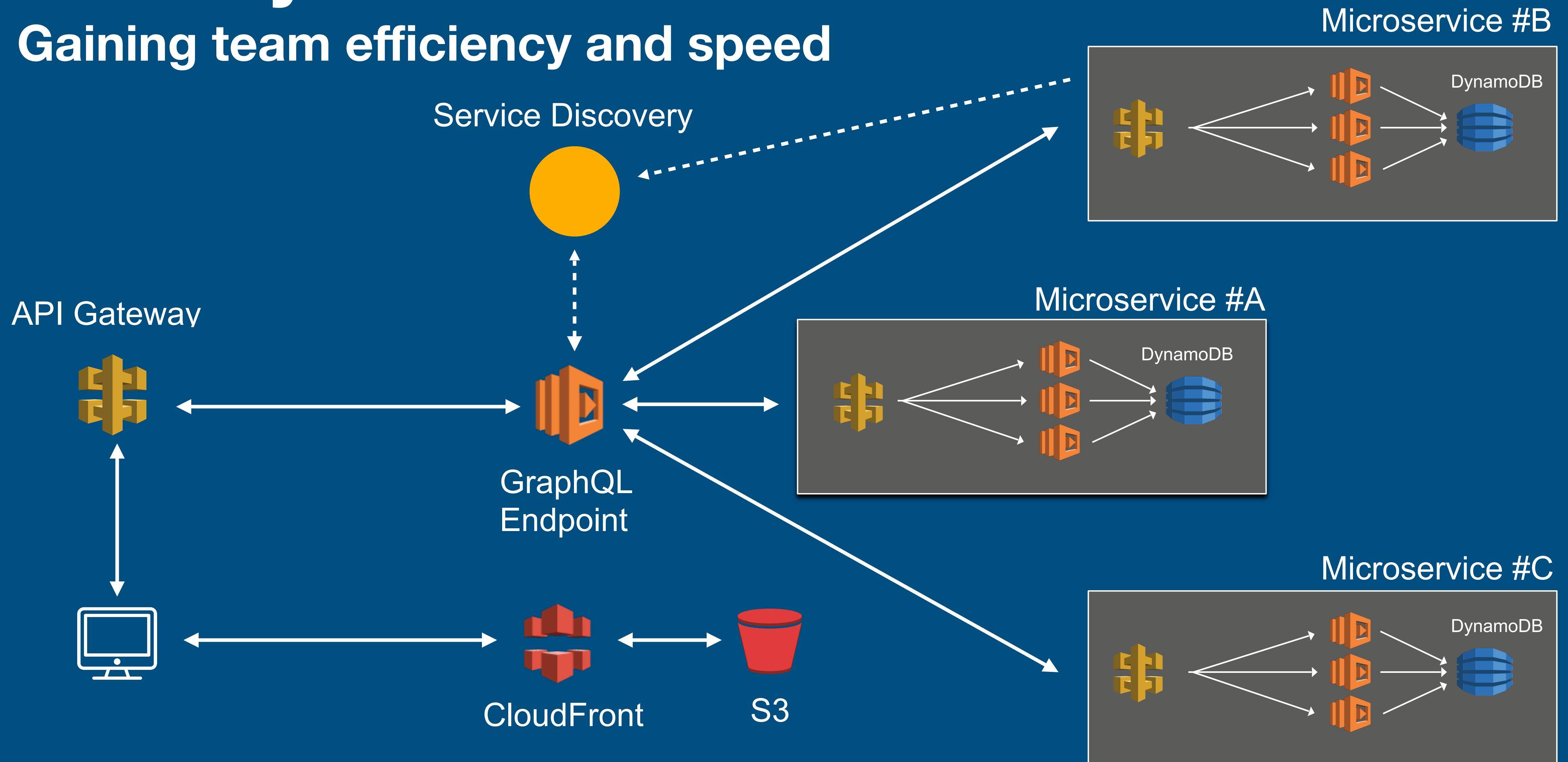
And the costs weren't too bad either

- 289 Lambda Functions
- 19 Micro-services
- 3.68TB of data in S3
  
- 107m Lambda Invocations / month
- 45m API Requests / month
- 3.8+ TB of data via CloudFront per day
- 650K+ users

Service	Cost
Key Management Service	\$25.26
Simple Storage Service (S3)	\$108.23
Config	\$109.84
Elastic Transcoder	\$154.17
<b>API Gateway</b>	<b>\$192.14</b>
Developer Support	\$314.59
Redshift	\$371.50
DynamoDB	\$373.54
<b>Lambda</b>	<b>\$706.49</b>
CloudWatch	\$3,142.73
CloudFront	\$5,099.85

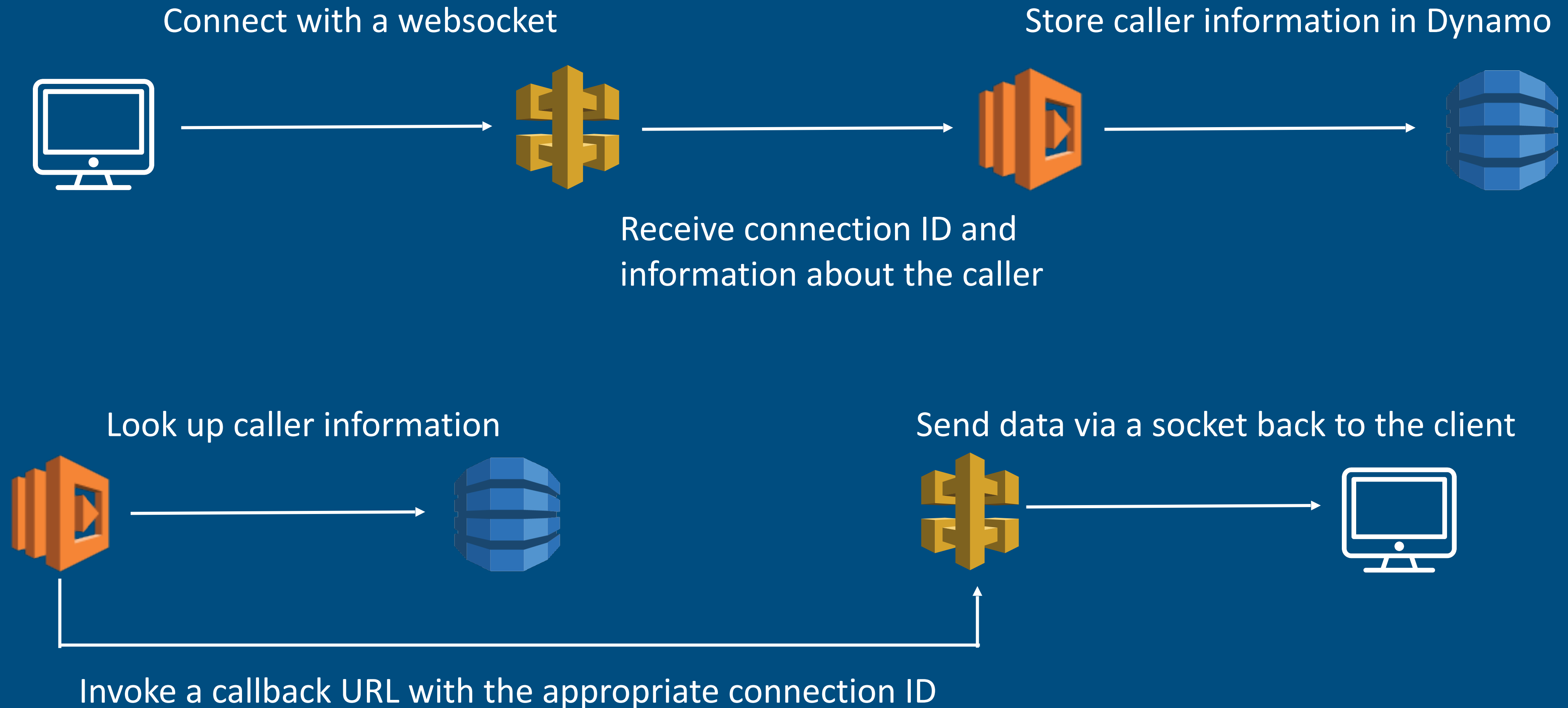
# Maturity

Gaining team efficiency and speed





# Web Sockets With API Gateway





# Mullet Architecture

With thanks to Tim Wagner





# Common Benefits

## When things go right

- It's fast to build (shortened time to market)
- Massive scale and initially can be cheap or free
- It's operationally efficient
- It's not Kubernetes
- Large architectural pivots/changes are possible
- It's fun - developers love it

# Common Complaints

## Why can't things just be easy

- Hard to dev locally
- Hard to debug
- Hard to observe and monitor
- Hard or impossible to do certain things (e.g. long-running tasks)
- Lock-in is a problem. Maybe?



# How I build today

## Back to working on my own

- Back to building on my own
- Serverless first approach makes technical decisions easier
- Have to be fast and reduce operational overhead to zero
- Could potentially need a lot of compute down the road
- Leverage as many AWS services as possible
- Solve local dev & debugging

# fatfireapp.com

## Wealth Operating System

fatfireapp.com:3000/portfolio/0a05f3a0/assets/a1e140bb

AUD Portfolio Test 2 Peter Sbarski

Link to Bank Refresh Data Rename Currency Remove

Asset	Value
Checking Account	AUD 90,892.32
Credit Card	AUD 6,972.71
Credit Card 13000	AUD 3,537.64
Hooli Access	AUD -25,194.11
Hooli Awards Card	AUD -4,246.62
Hooli Saver	AUD -20,510.14
Loan for vacation 2017	AUD 5,521.92
Main	AUD 22,397.61

fatfireapp.com:3000/portfolio/0a05f3a0

fatfire

Dashboard

YOUR FINANCIALS

- Bank Accounts
- Shares & ETFs
- Property

NET WORTH

- Net Worth
- FIRE
- EXTRAS
- Transactions
- Insurance
- Receipts
- Taxation

Portfolio Test 2

Welcome Peter Sbarski, everything seems great!

Track important

**\$1,265,594.4**  
ASSETS

**-\$241,560.68**  
LIABILITIES

ASSET

ASSET	CATE
<u>Plaid Checking</u> Plaid Accounts	Bank
<u>Online</u> ING	Bank
<u>Orange One Low Rate Platinum</u> ING	Bank
<u>Superannuation</u> ING	Bank

fatfireapp.com:3000/portfolio/0a05f3a0/worth

AUD Portfolio Test 2 Peter Sbarski

Dashboard

YOUR FINANCIALS

- Bank Accounts
- Shares & ETFs
- Property

NET WORTH

- Net Worth
- FIRE
- EXTRAS
- Transactions
- Insurance
- Receipts
- Taxation

**\$1,265,594.4** -0.12%  
ASSETS

**-\$241,560.68** 0.60%  
LIABILITIES

**\$1,024,033.72** -0.28%  
NET WORTH

Portfolio History

Portfolio Assets Debts

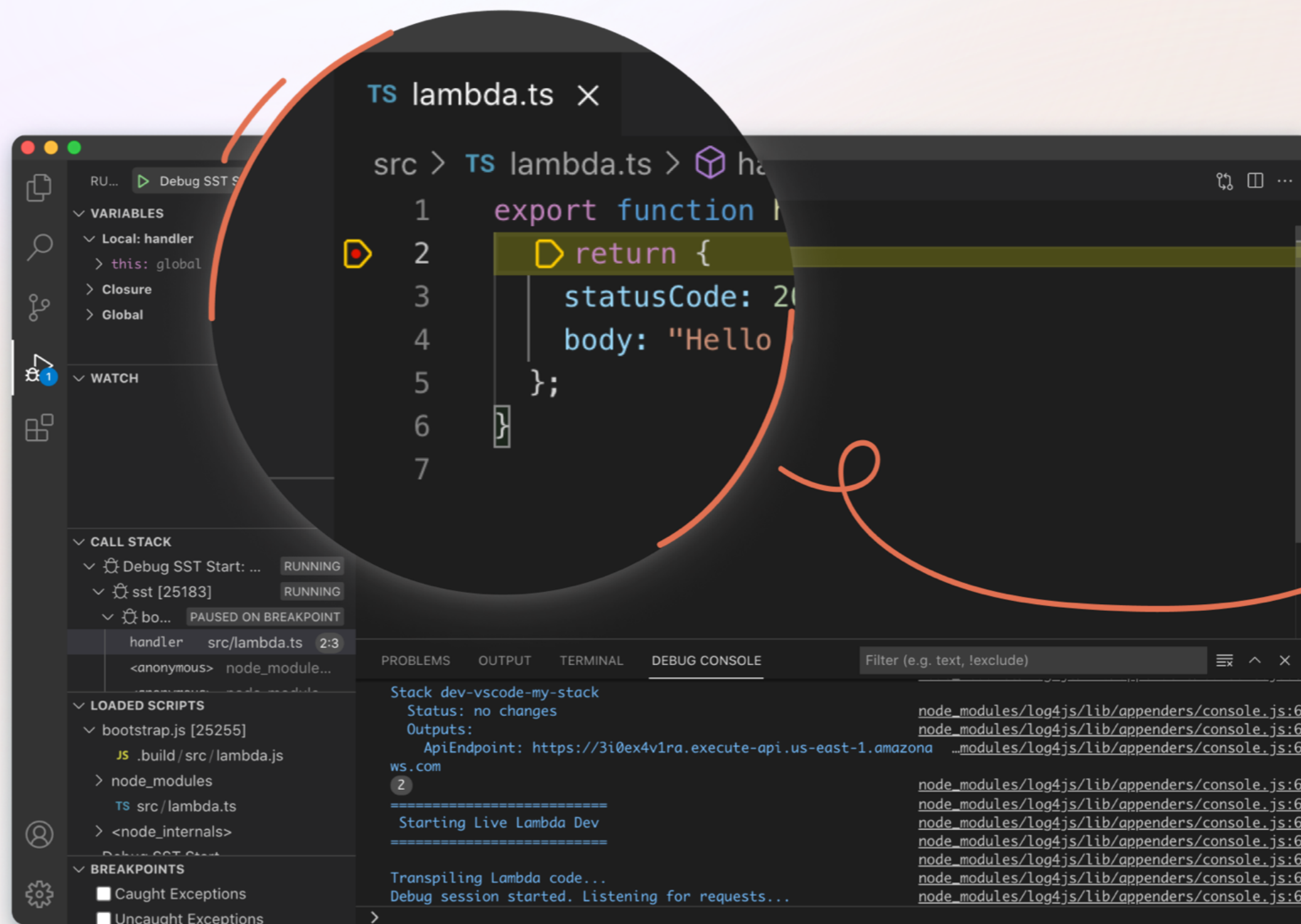


# Serverless Stack (SST)

<https://sst.dev>

## Test your apps live

Set breakpoints in your Lambda functions and test your apps live. [Learn more >](#)



*Set breakpoints in your functions*

The screenshot shows the SST web interface. On the left is a sidebar with navigation items: Local, Stacks, Functions, API, DynamoDB, RDS, Buckets, GraphQL, and Cognito. The main area displays a list of 'Invocations' for a function named 'pete-dev'. Each invocation entry includes a status (e.g., 'Success'), a timestamp, and a truncated log snippet.

Status	Timestamp	Log Snippet
Success	06:57:37.368	GET /integration/transactions ▶ "Request" : {...} 8 items
	06:57:37.738	Missing transactions for [] getting cached response false
	06:57:38.216	▶ "Response" : {...} 3 items
Success	06:57:37.306	GET /integration/transactions ▶ "Request" : {...} 8 items
	06:57:37.736	Missing transactions for [] getting cached response false
	06:57:38.200	▶ "Response" : {...} 3 items
Success	06:57:37.303	GET /integration/transactions ▶ "Request" : {...} 8 items
	06:57:37.738	Missing transactions for [] getting cached response false
	06:57:38.215	▶ "Response" : {...} 3 items
Success	06:57:37.300	GET /integration/transactions ▶ "Request" : {...} 8 items
	06:57:37.737	Missing transactions for [] getting cached response false
	06:57:38.216	▶ "Response" : {...} 3 items

# Serverless Stack

## Quick overview

- Local dev & debugging
- Built on top of CDK
- Console for administration of your resources
- Higher constructs like Function and Config
- Auth (new!)

The screenshot shows the SST console interface. On the left is a navigation menu with the following items:

- Local
- Stacks
- Functions
- API
- DynamoDB
- RDS
- Buckets
- GraphQL
- Cognito

The right side of the console displays a log viewer with the following entries:

- Error** TransactionFailCatch  
12:00:31.110 ▶ "Request" : { ... } 2 ...
- 12:00:31.113 {  
Error: 'Lambda.Unknown  
Cause: 'The cause cou  
07T12:00:30.744Z db7f3a  
}
- 12:00:31.114  
SyntaxError: Unexpected  
at JSON.parse (<anonymo  
at Runtime.exports.hand  
TransactionFailCatch/sr  
at Runtime.handleOnce (C  
ric/lib/Runtime/Runtime  
at processTicksAndRejec
- Error** TransactionFailCatch  
12:00:30.597 ▶ "Request" : { ... } 2 ...
- 12:00:30.598 {  
Error: 'Lambda.Unknown  
Cause: 'The cause cou  
07T12:00:30.320Z 652ab2  
}
- 12:00:30.598  
SyntaxError: Unexpected  
at JSON.parse (<anonymo



# Serverless Stack - Auth

## This is new

- Main components:
  - Auth - a construct that creates the necessary infrastructure
  - AuthHandler - a Lambda handler function that can handle authentication flows for various providers
  - Session - a library for issuing and validating authentication sessions in your Lambda function code.
- Source: <https://docs.sst.dev/auth>

Authenticate through third-party providers

and/or send magic links

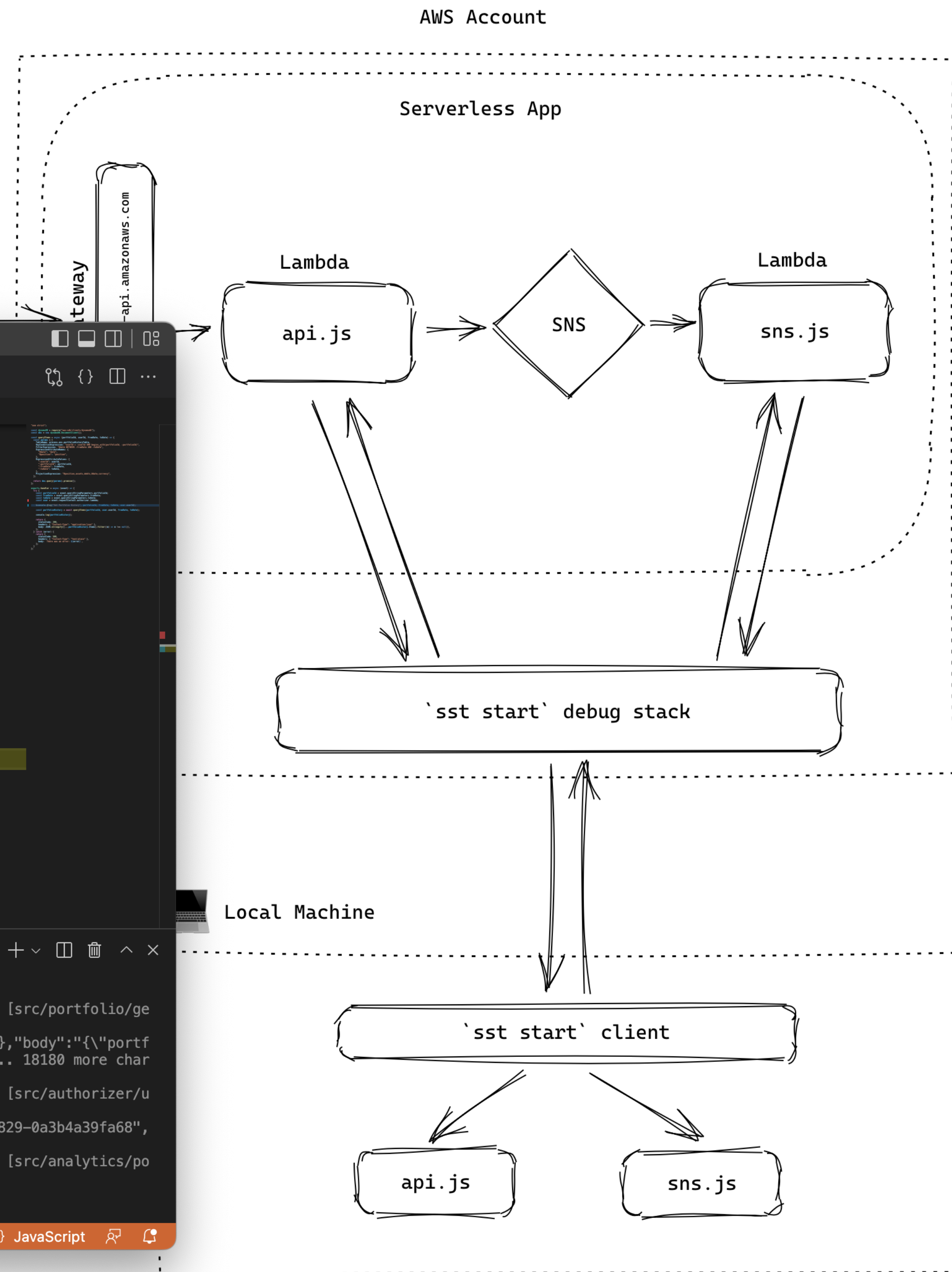
Don't store or implement password functionality

# Serverless Stack (SS) Local Dev. Absolutely Brilliant.

```
src > analytics > portfolio > JS lambda.js > handler > handler
19     ":toDate": toDate,
20   },
21   ProjectionExpression: "#position,assets,debts,#date,currency",
22 };
23
24   return doc.query(params).promise();
25 };
26
27   exports.handler = async (event) => {
28     try {
29       const portfolioId = event.queryStringParameters.portfolioId;
30       const fromDate = event.queryStringParameters.fromDate;
31       const toDate = event.queryStringParameters.toDate;
32       const user = event.requestContext.authorizer.lambda;
33
34       console.log("Get Portfolio History", portfolioId, fromDate, toDate, user.userId);
35
36       const portfolioHistory = await queryItems(portfolioId, user.userId, fromDate, toDate);
37
38       console.log(portfolioHistory);
39
40       return {
41         statusCode: 200,
```

DEBUG CONSOLE

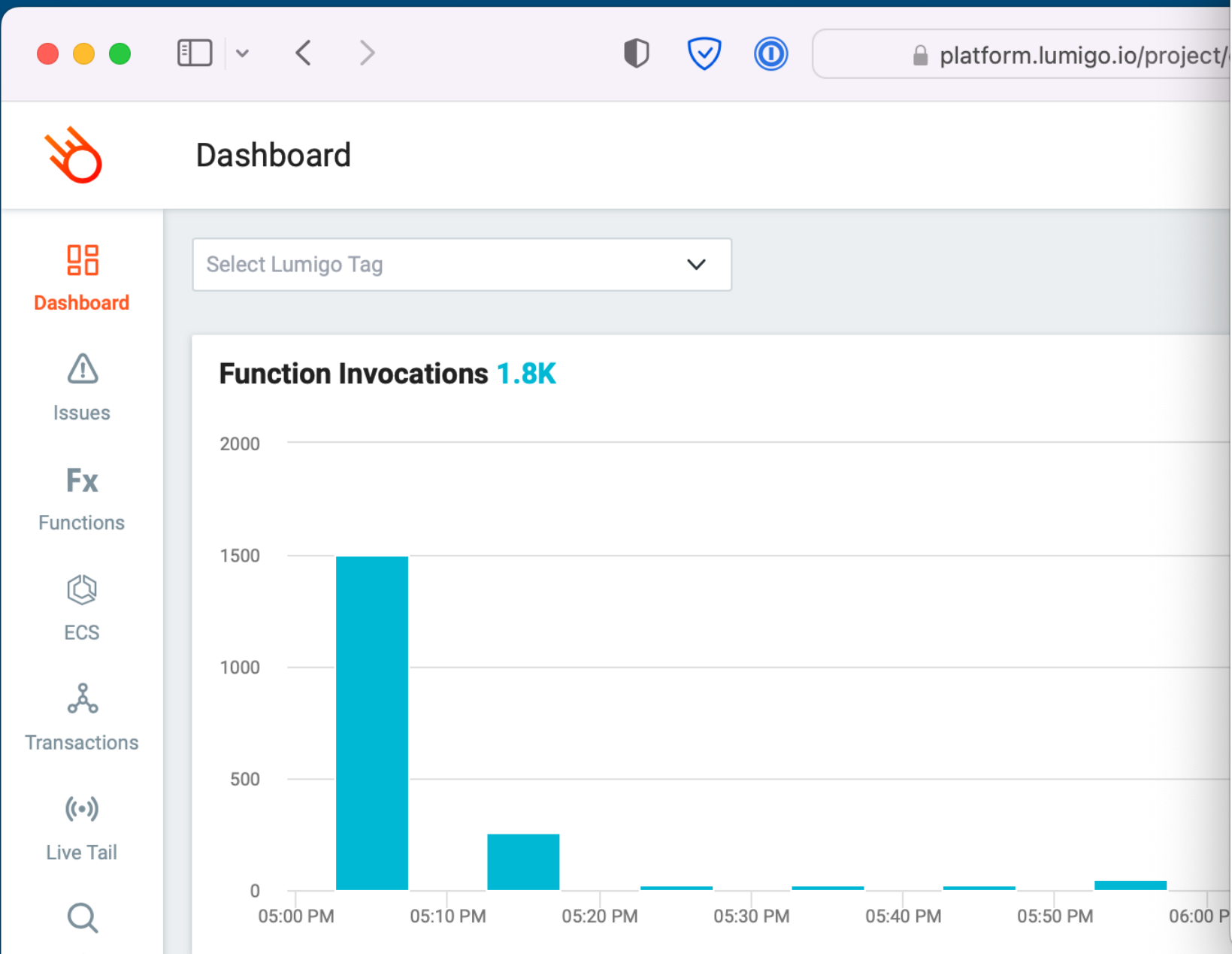
```
"tenantId":"rbv68z4b"}
0749100b-d4c4-45b6-bbf8-0cbd9b4b21d7 REQUEST pete-dev-fat-fire-app-cor-ApiLambdaGETportfolioC4A-6sh6fu2MqSdH [src/portfolio/get/lambda.handler] invoked by API GET /portfolio
0749100b-d4c4-45b6-bbf8-0cbd9b4b21d7 RESPONSE {"statusCode":200,"headers":{"Content-Type":"application/json"},"body":{"portfolioId":"0a05f3a0","currency":"AUD","updatedAt":"2022-08-31T06:56:31.223Z","userId":"cd8ebcc0... 18180 more characters"}
1e0299bf-0120-40da-9d80-c7c6228ac147 REQUEST pete-dev-fat-fire-app-cor-UserAuthorizerFn8DFD7EFB-5Q9E8H8tkPZ2 [src/authorizer/user/lambda.handler] invoked by API GET /analytics/portfolio
1e0299bf-0120-40da-9d80-c7c6228ac147 RESPONSE {"isAuthorized":true,"context":{"userId":"cd8ebcc0-153b-4912-8829-0a3b4a39fa68","tenantId":"rbv68z4b"}}
595f68d3-abd6-4b81-8df2-cae42cf71b8c REQUEST pete-dev-fat-fire-app-ana-LambdaGETanalyticsportfo-9oX4Pugh81df [src/analytics/portfolio/lambda.handler] invoked by API GET /analytics/portfolio
Debugger attached.
```





# Observability

<https://lumigo.io>



### Most Failed Functions

Function Name	Lumigo ...	% Failur...	# Failures ↓
pete-dev-fat-fire-app-cor-... us-east-1	Not Set	4.12%	8
pete-dev-fat-fire-app-int-B... us-east-1	Not Set	100%	1
pete-dev-fat-fire-app-his-... us-east-1	Not Set	100%	1

### Transactions

Search by entry point  Transactions with issues only

Transactions	Result	Start Time	Duration	Cost	Issues
pete-dev-fat-fire-app-cor-ApiLambdaGETportfolioC4A-6sh6fu... us-east-1	Success	5:55:43 PM	1,886 ms	< 1¢	-
pete-dev-fat-fire-app-cor-ApiLambdaGETportfolioC4A-6sh6fu... us-east-1	Success	5:54:38 PM	937 ms	< 1¢	-
pete-dev-fat-fire-app-cor-ApiLambdaGETportfolioC4A-6sh6fu... us-east-1	Success	5:54:12 PM	2,568 ms   ❄️ 1	< 1¢	-
pete-dev-fat-fire-app-his-ProcessUsersTimezoneC1DD-NAujh... us-east-1	Failure	5:51:13 PM	1,993 ms   ❄️ 1	< 1¢	Runtime.UnhandledPromiseRejection
pete-dev-fat-fire-app-int-BasiqTokenRotation825450-QGbWW... us-east-1	Failure	5:30:17 PM	2,199 ms   ❄️ 1	< 1¢	Runtime.UnhandledPromiseRejection
pete-dev-fat-fire-app-cor-UserAuthorizerFn8DFD7EFB-5Q9E8... us-east-1	Failure	5:06:40 PM	4 ms	< 1¢	Error
pete-dev-fat-fire-app-cor-UserAuthorizerFn8DFD7EFB-5Q9E8... us-east-1	Failure	5:06:40 PM	1 ms	< 1¢	Error
pete-dev-fat-fire-app-cor-UserAuthorizerFn8DFD7EFB-5Q9E8... us-east-1	Failure	5:06:40 PM	3 ms	< 1¢	Error
pete-dev-fat-fire-app-cor-UserAuthorizerFn8DFD7EFB-5Q9E8... us-east-1	Failure	5:06:40 PM	1 ms	< 1¢	Error

1 to 14 of 14 Page 1 of 1

### Most Invoked Functions

Function Name	Lumigo Tag	# Invocations ↓
pete-dev-fat-fire-app-debug-stac... us-east-1	Not Set	772
pete-dev-fat-fire-app-debug-sta-D... us-east-1	Not Set	507
pete-dev-fat-fire-app-cor-UserAut... us-east-1	Not Set	194
pete-dev-fat-fire-app-int-Lambda... us-east-1	Not Set	172
pete-dev-fat-fire-app-debug-stac... us-east-1	Not Set	157

# Serverless Cloud

<https://www.serverless.com/cloud>

## Just. Write. Code.

Serverless Cloud lets you build full stack applications better and faster than anyone else using our innovative **Infrastructure FROM Code**. Create zero-config backends in seconds along with frontend frameworks like React, Vue.js, SvelteKit, Next.js, and 11ty, all in a unified developer experience.

Get Started for Free

```
Text Editor
const { api } = require("@serverless/cloud")
api.get("/todos", async (req, res) => {
  let result = await data.get("todos:*");
  res.send({ items: result.items })
})

Terminal
$ ccloud
✓ Connected to my-app on http://localhost:9000
⚡ > 0s > Deploying
```





# Fatfire Architecture



Core Service



Exchange Service



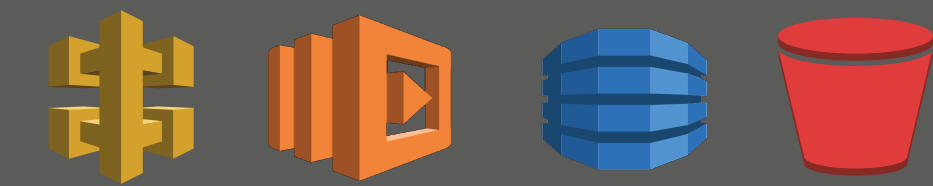
History Service



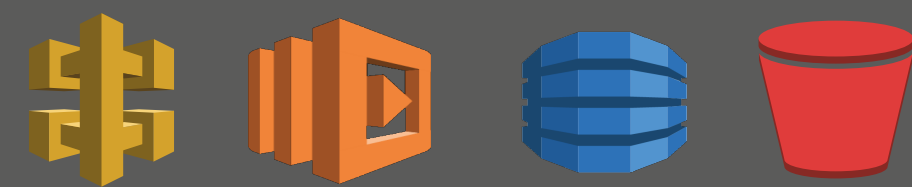
Cleanup Service



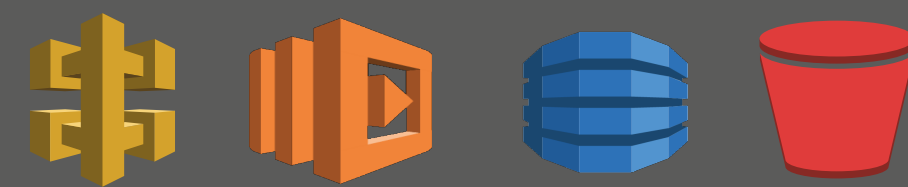
Analytics Service



Integration Service



Insurance Service

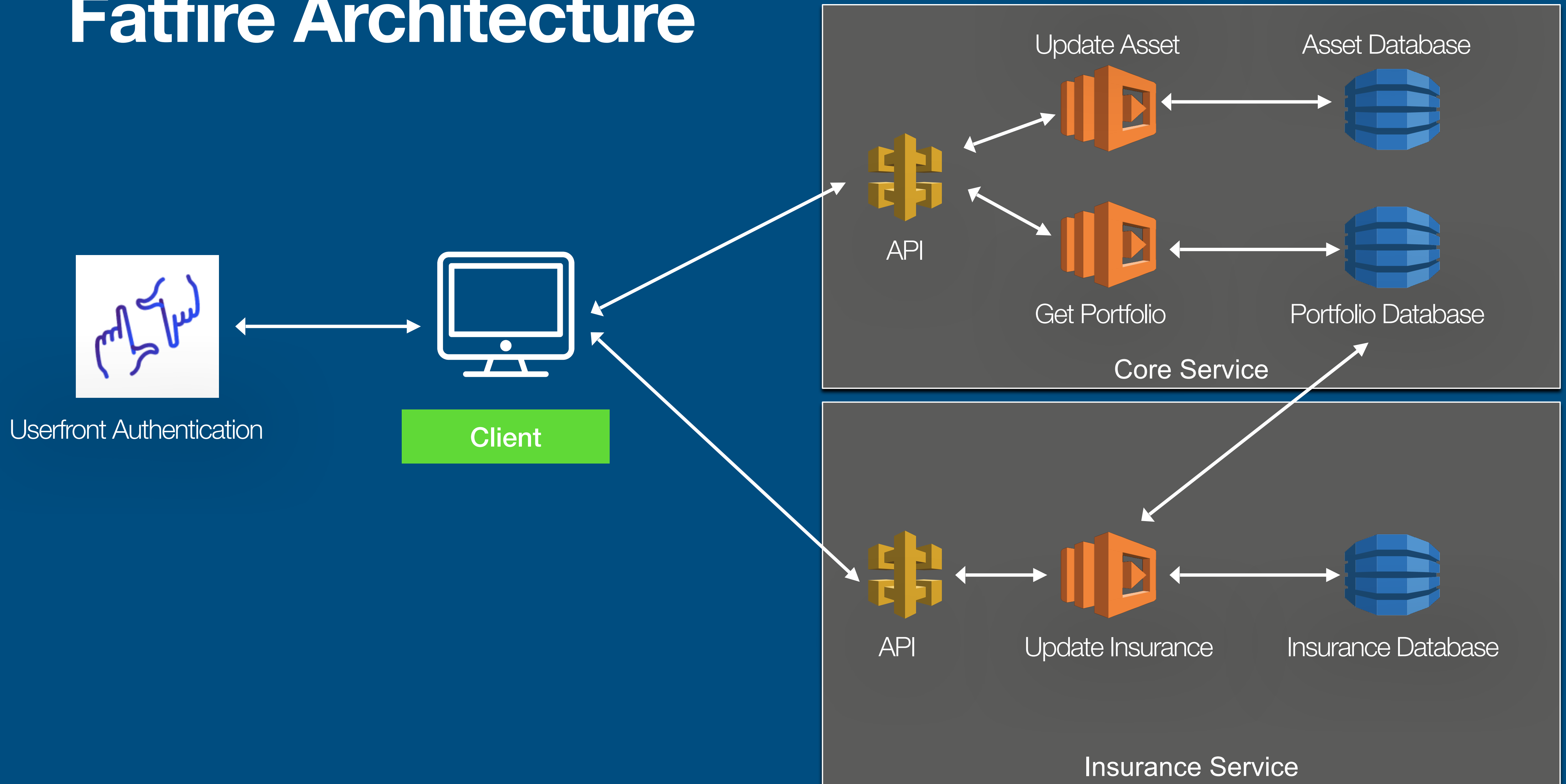


Receipts Service

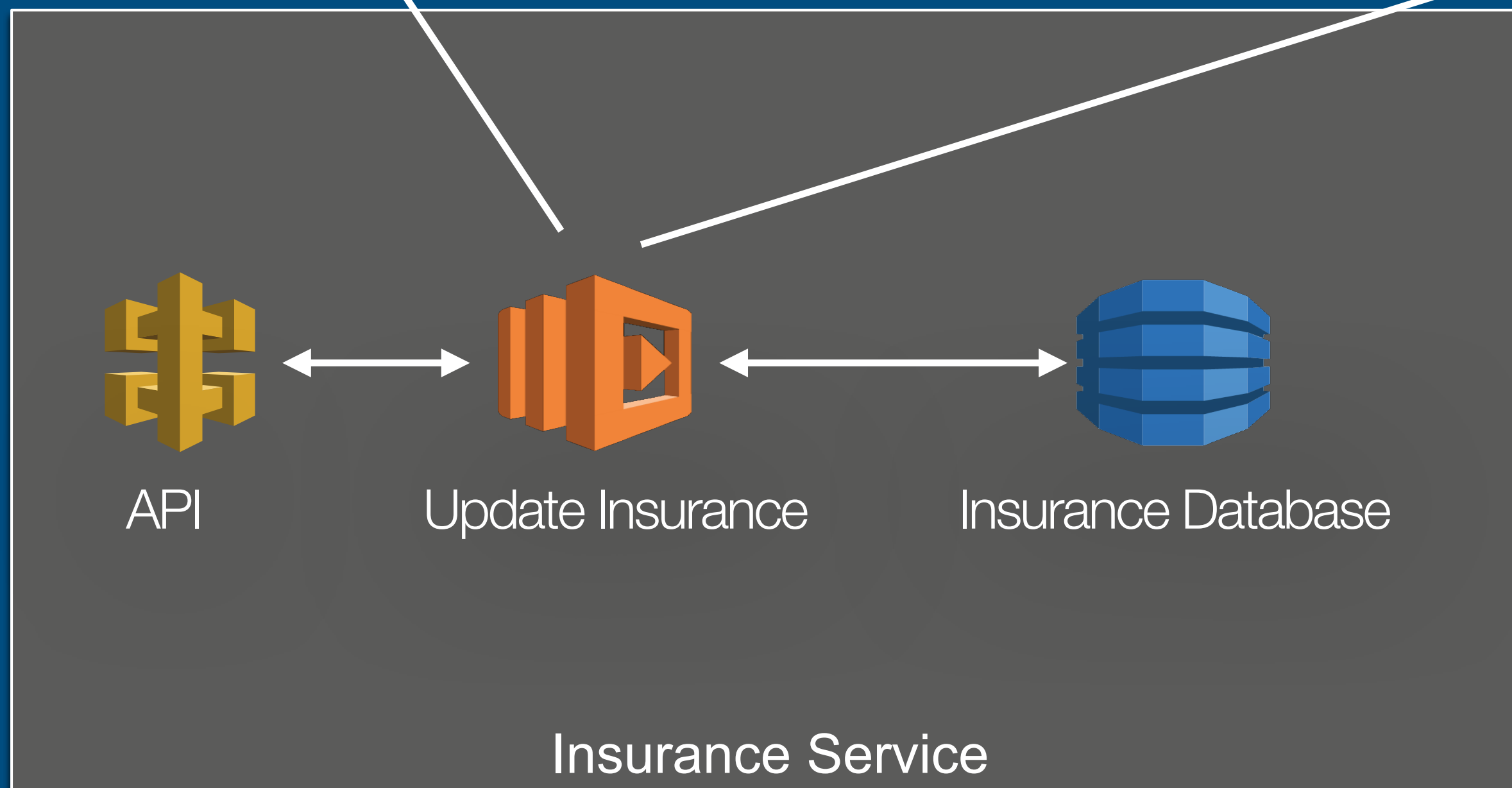
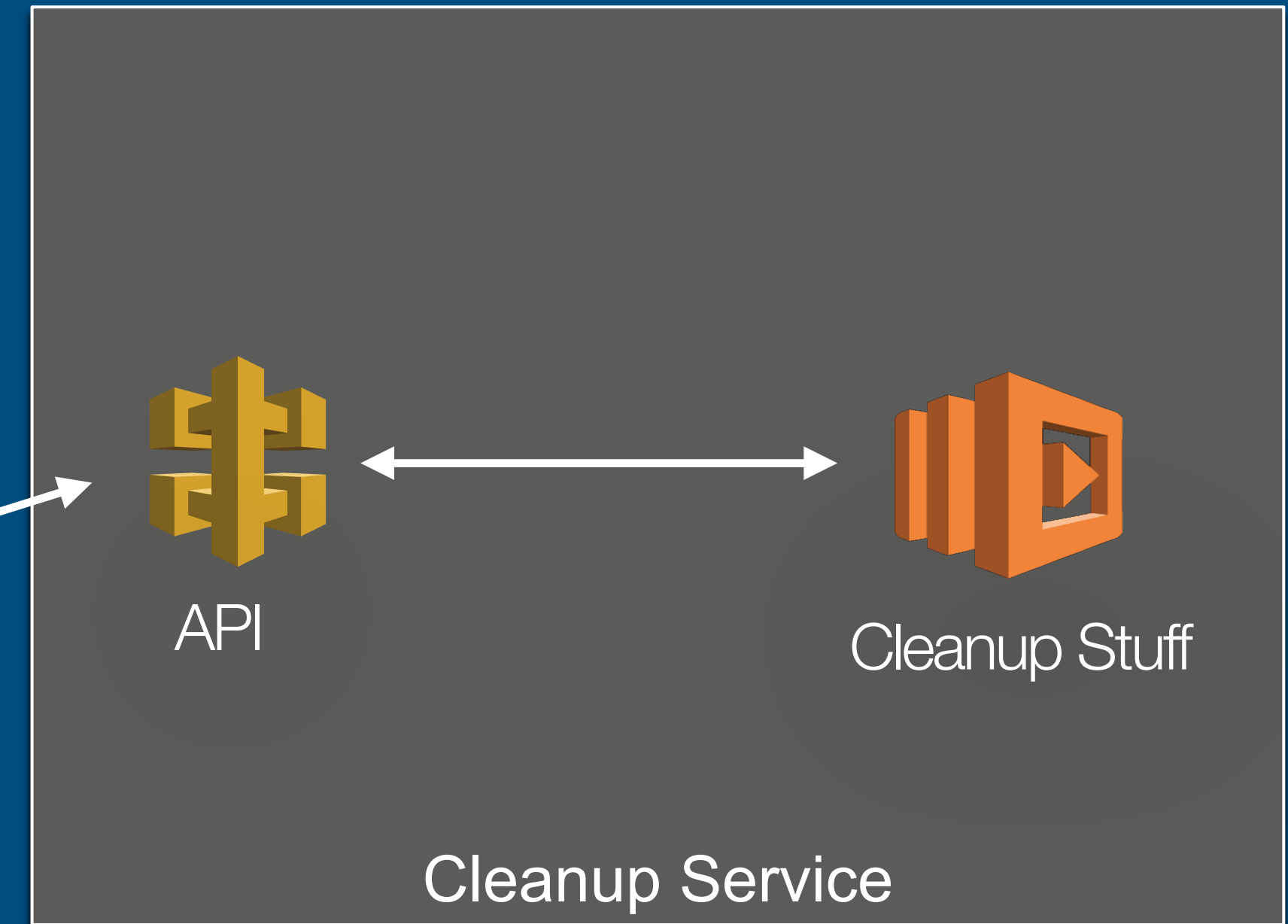
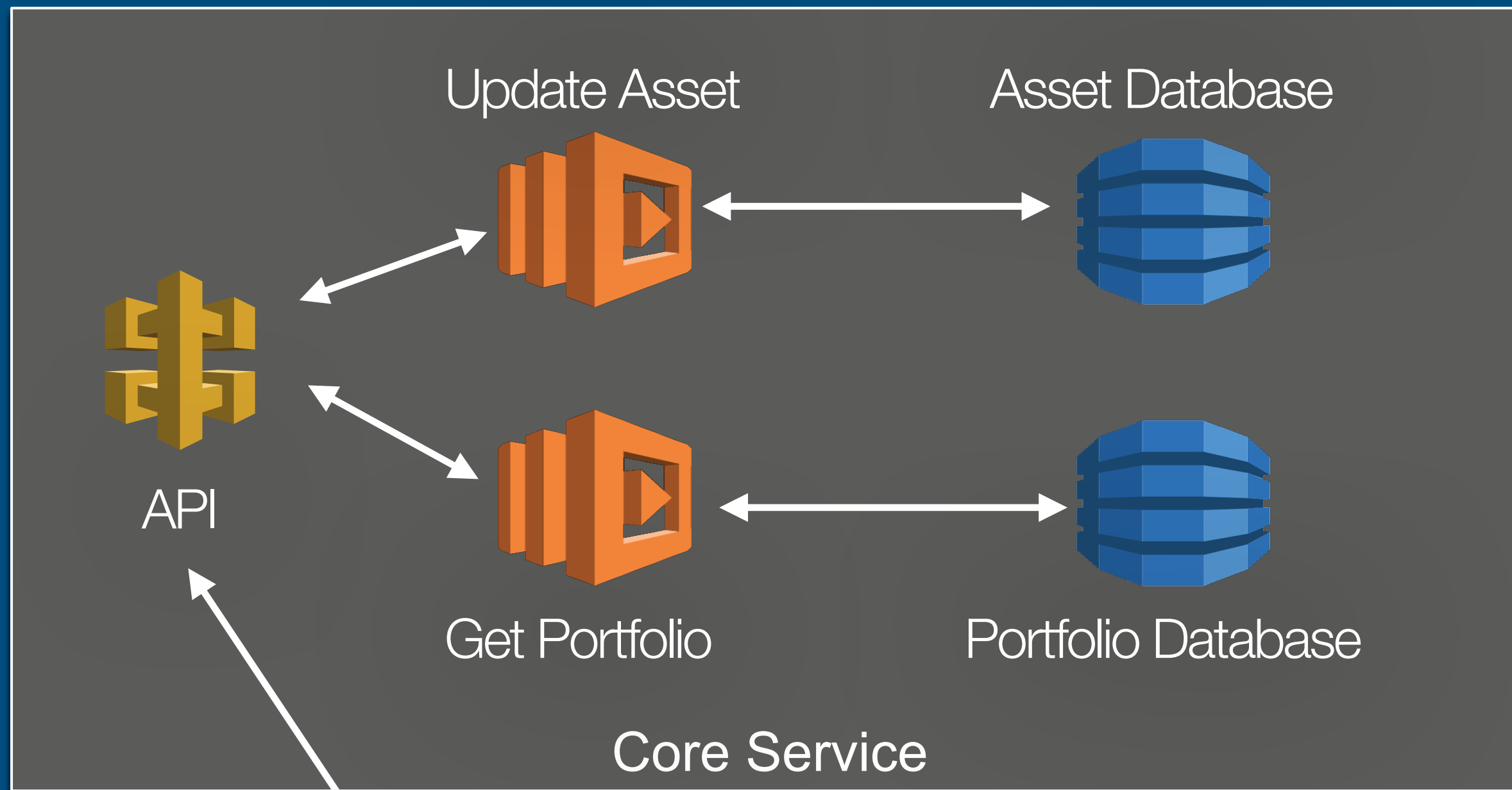


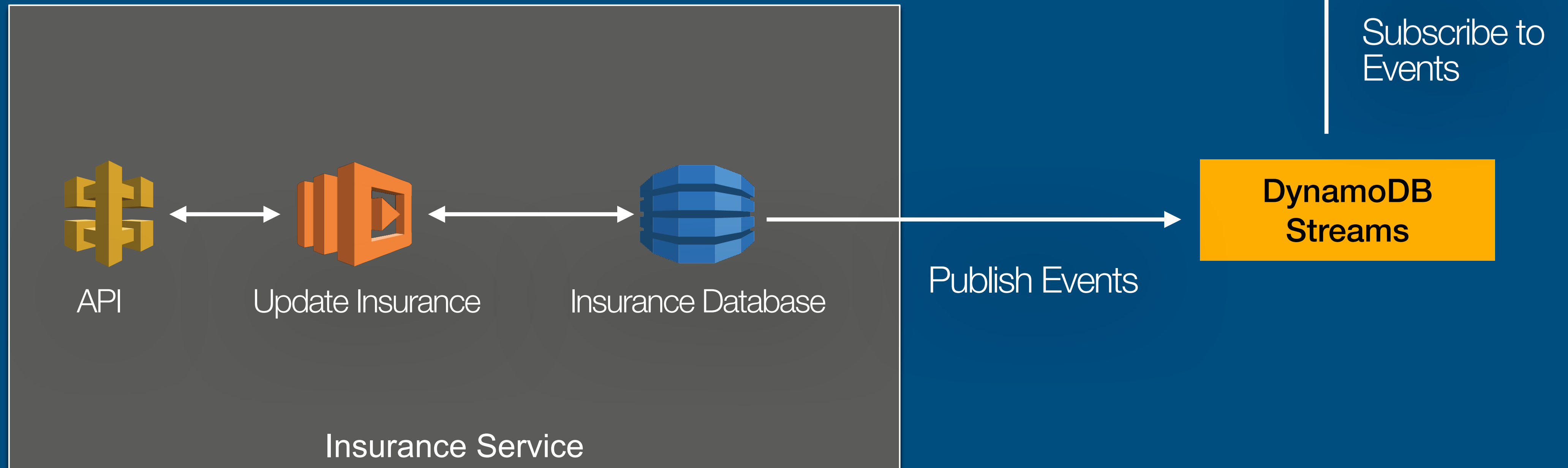
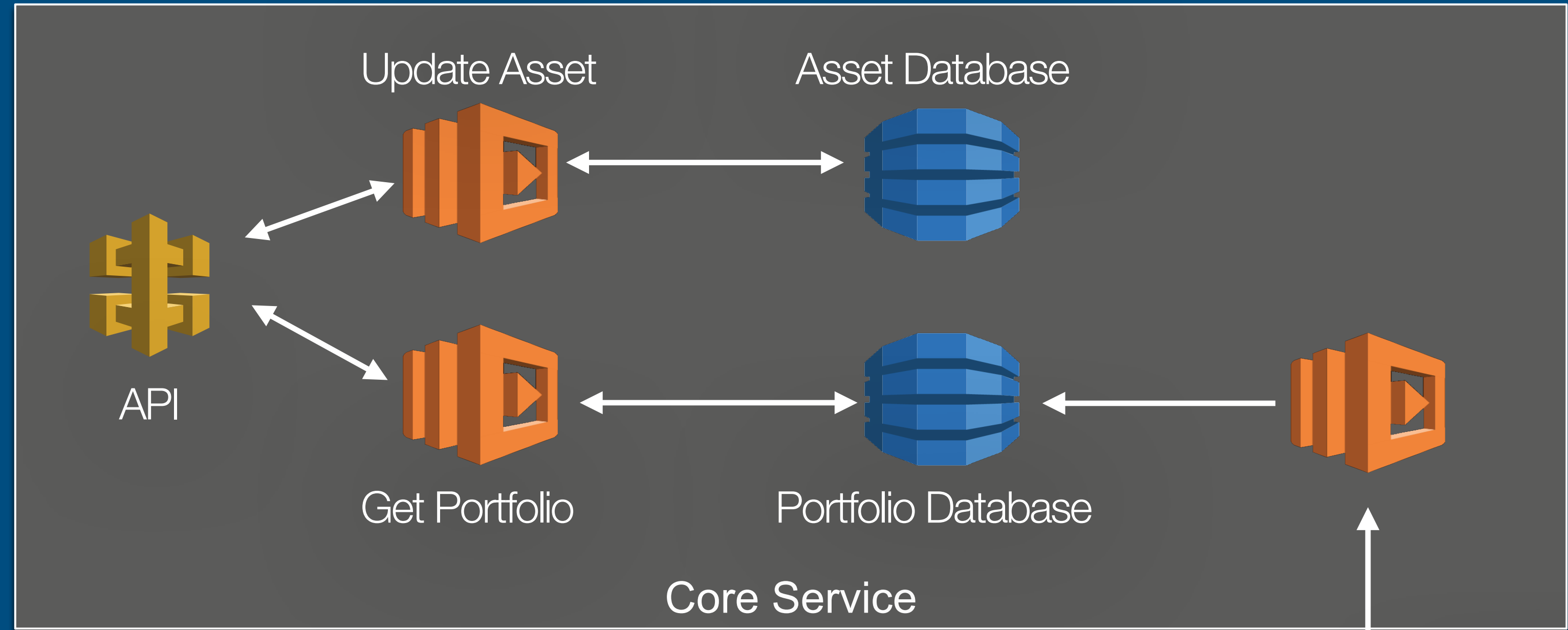
Tax Service

# Fatfire Architecture



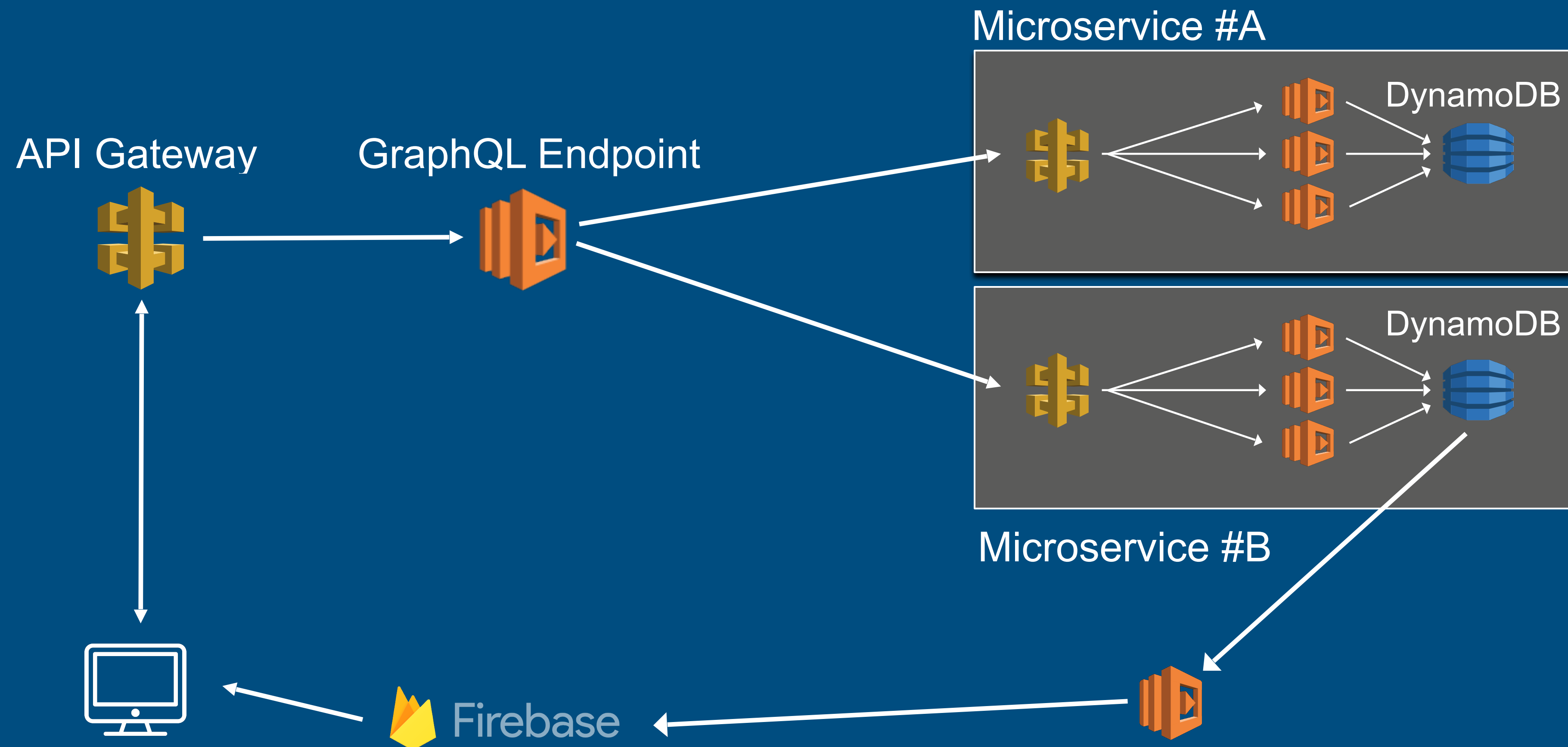


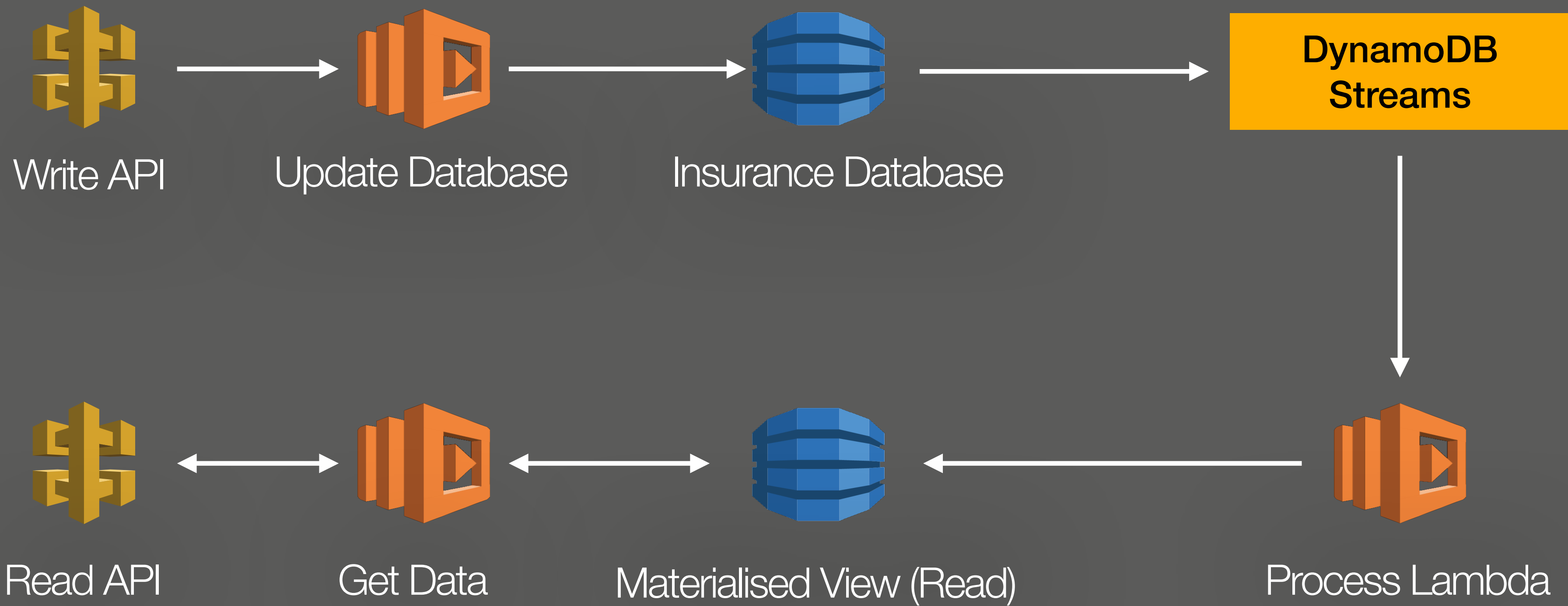




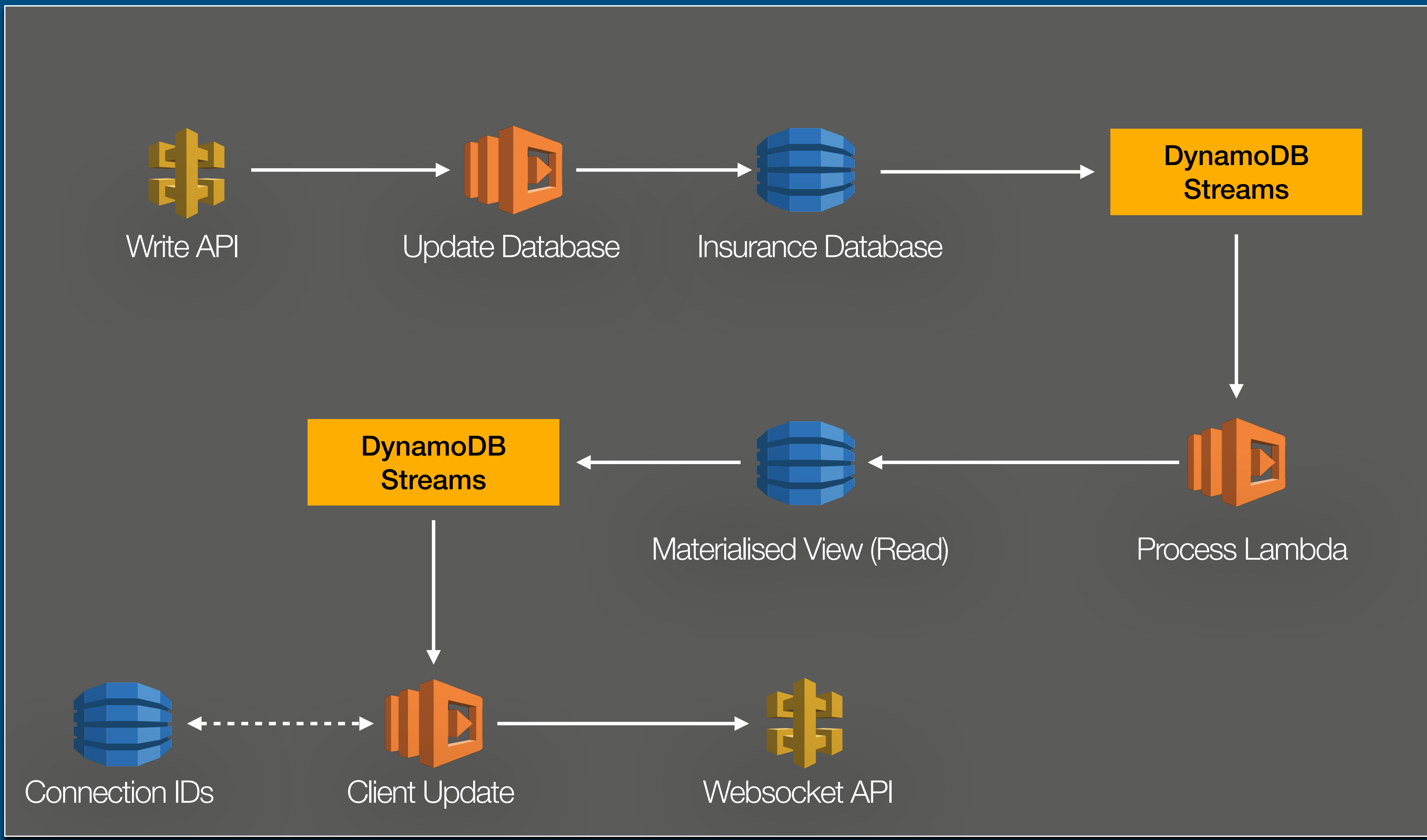


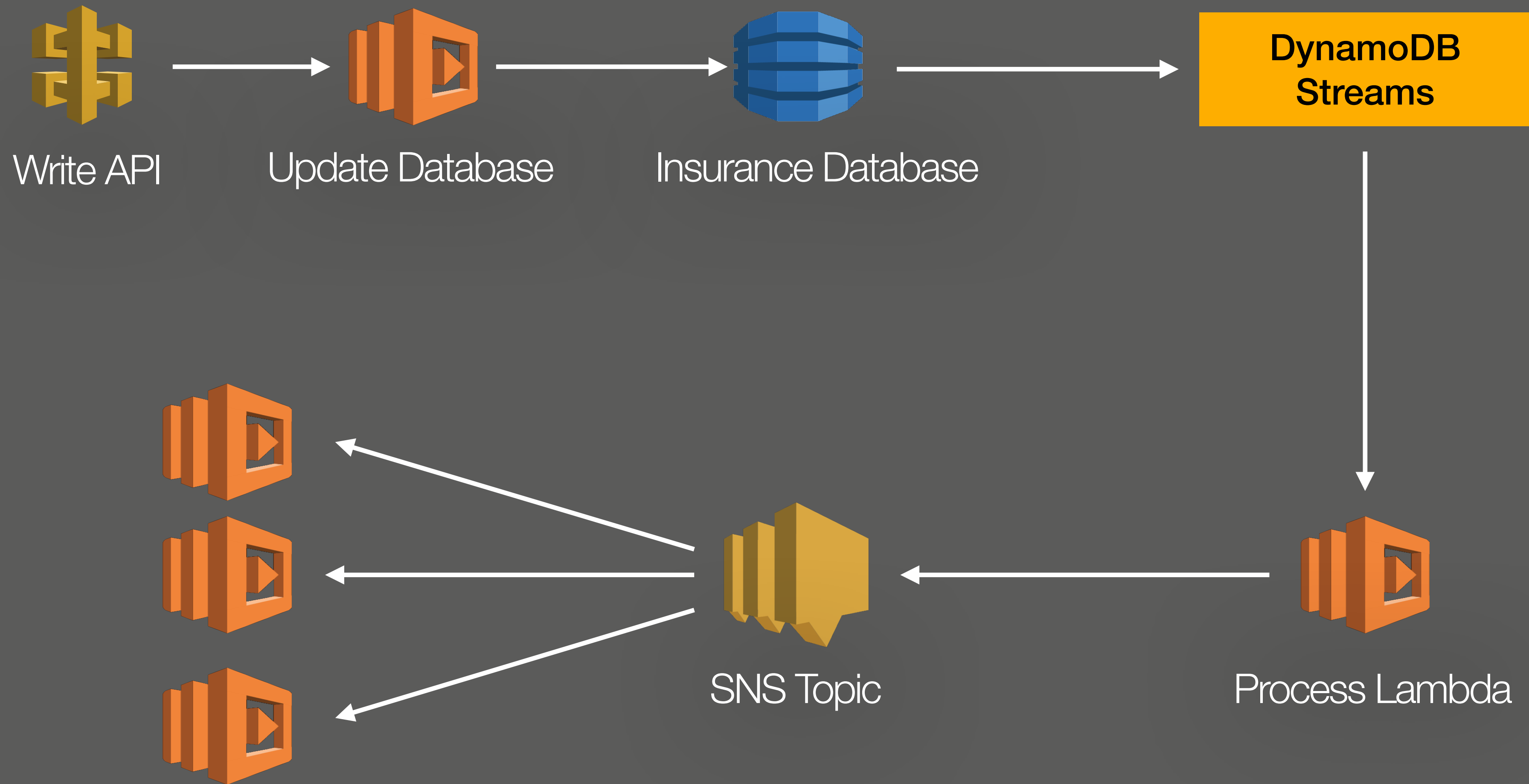
# Remember this architecture?



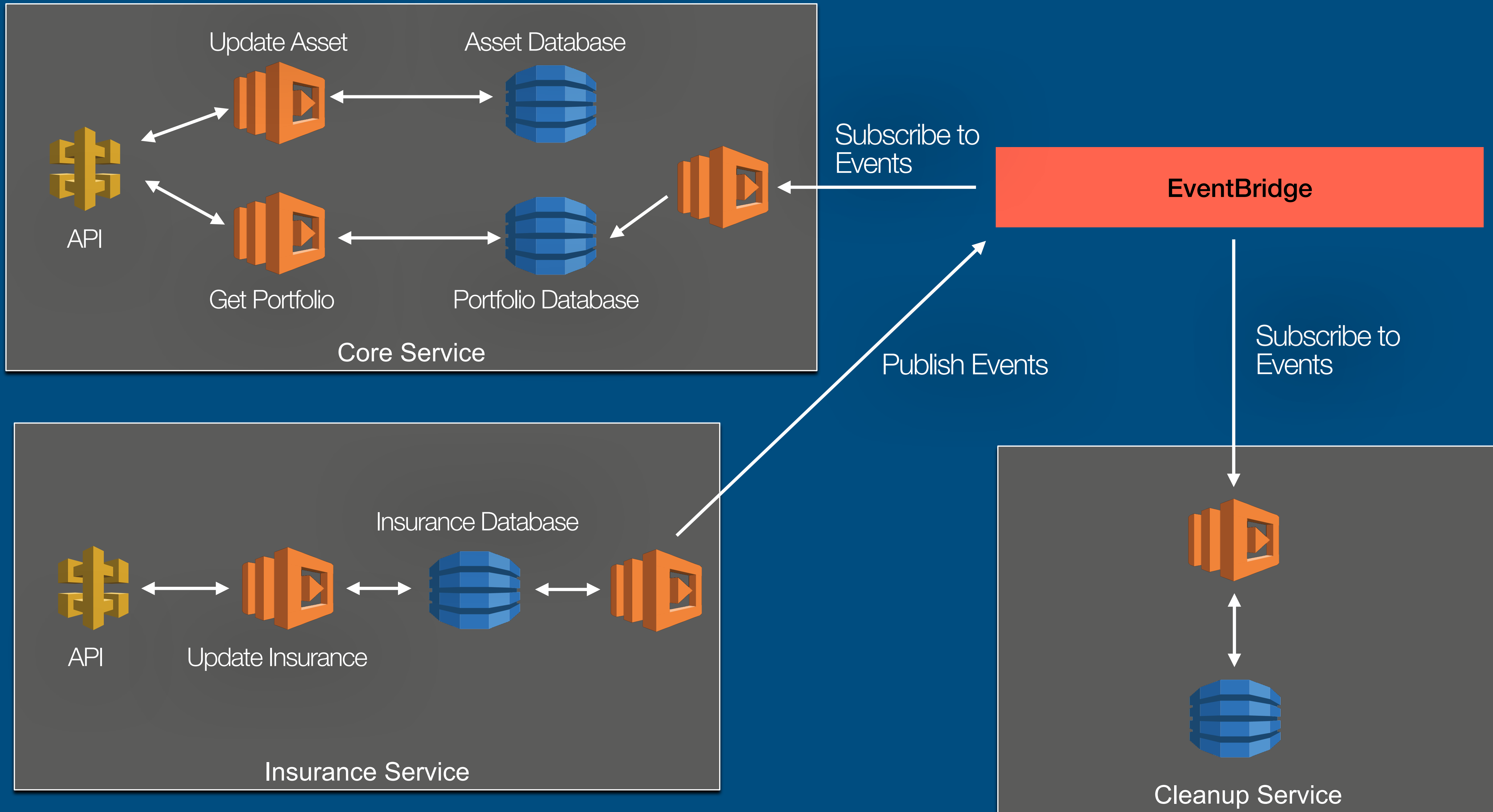












# Step Functions

## Coordinate Distributed Applications

Graph inspector

Data flow simulator [↗](#) Export [▼](#) Layout [▼](#)

+  
-  
⊙  
⊠

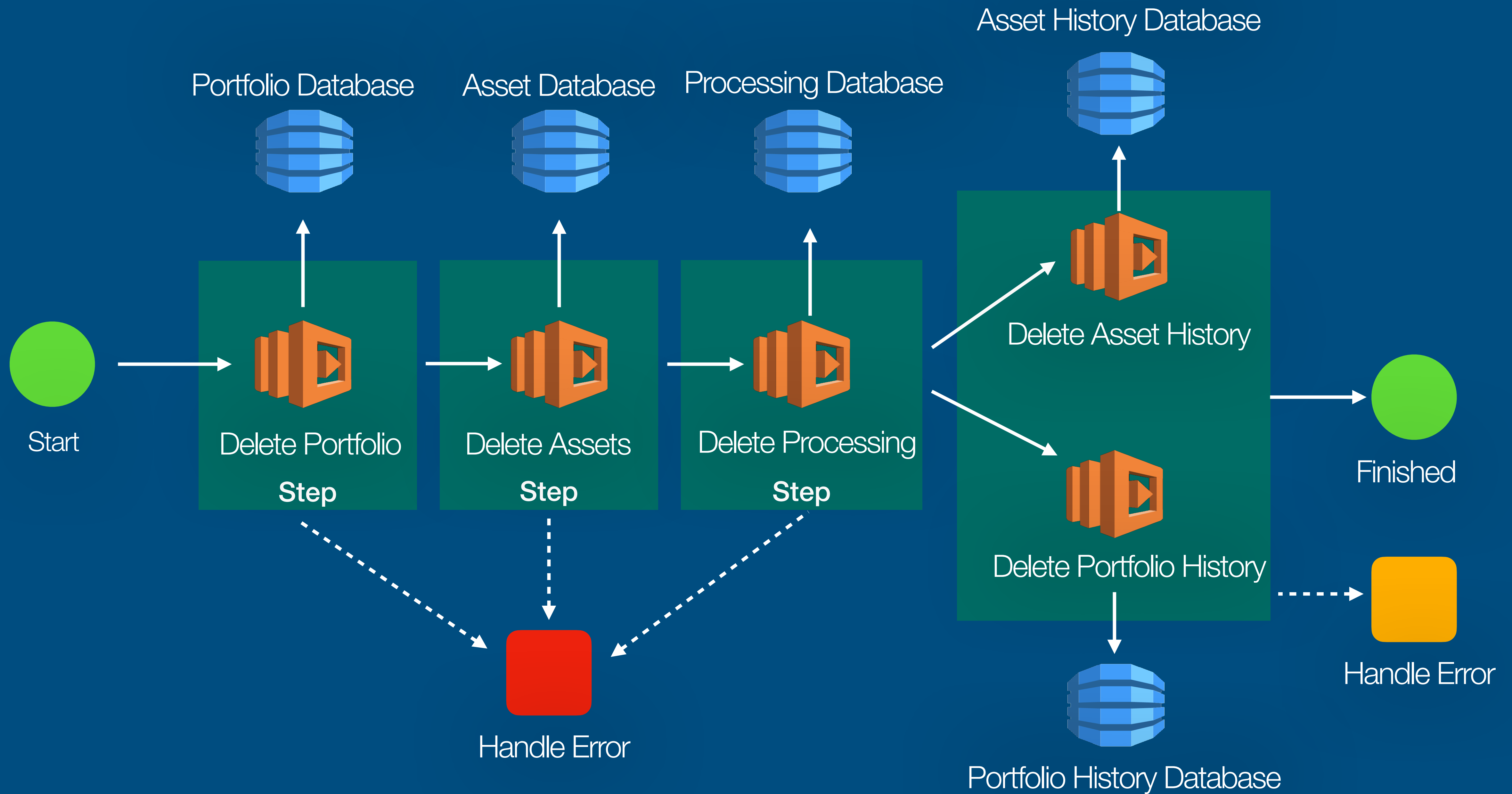
```
graph TD; Start((Start)) --> sDeletePortfolio[sDeletePortfolio]; sDeletePortfolio --> sDeleteAssets[sDeleteAssets]; sDeleteAssets --> sDeleteProcessing[sDeleteProcessing]; sDeleteProcessing --> sDeleteAssetHistory[sDeleteAssetHistory]; sDeleteProcessing --> sDeletePortfolioHistory[sDeletePortfolioHistory]; sDeleteAssetHistory --> PortfolioDeleted[Portfolio Deleted]; sDeletePortfolioHistory --> PortfolioDeleted; PortfolioDeleted --> End((End));
```

■ In Progress ■ Succeeded ■ Failed ■ Cancelled ■ Caught Error

Details | Step input | **Step output**

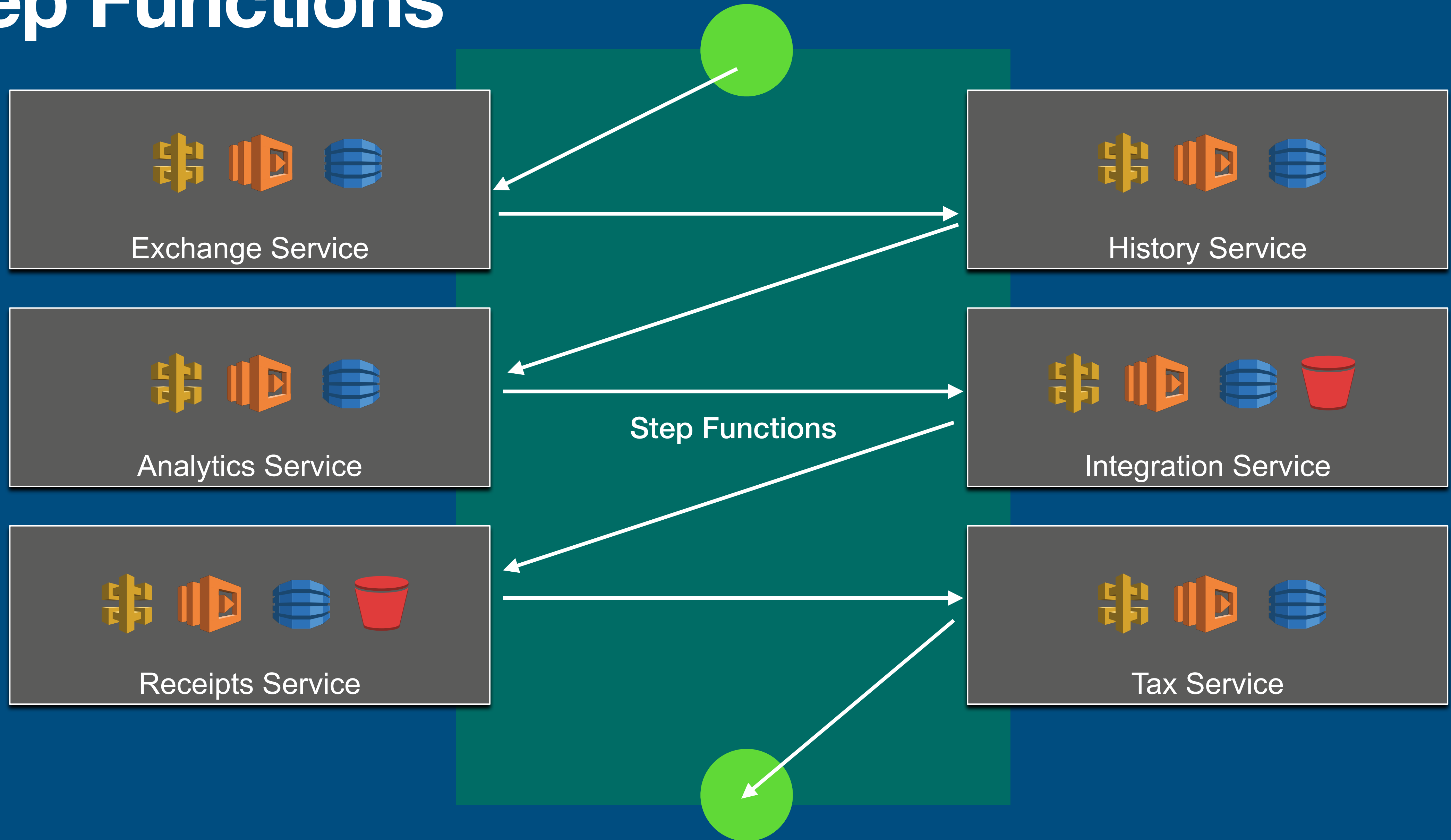
```
1 {
2   "ExecutedVersion": "$LATEST",
3   "Payload": {
4     "userId": "cd8ebcc0-153b-4912-8829-0a3b4a39fa68",
5     "portfolioId": "8e4b9a4b"
6   },
7   "SdkHttpMetadata": {
8     "AllHttpHeaders": {
9       "X-Amz-Executed-Version": [
10        "$LATEST"
11      ],
12      "x-amzn-Remapped-Content-Length": [
13        "0"
14      ],
15      "Connection": [
16        "keep-alive"
17      ],
18      "x-amzn-RequestId": [
19        "f1ee3ecd-7508-4462-bd83-56c43b58fhcd"

```





# Step Functions



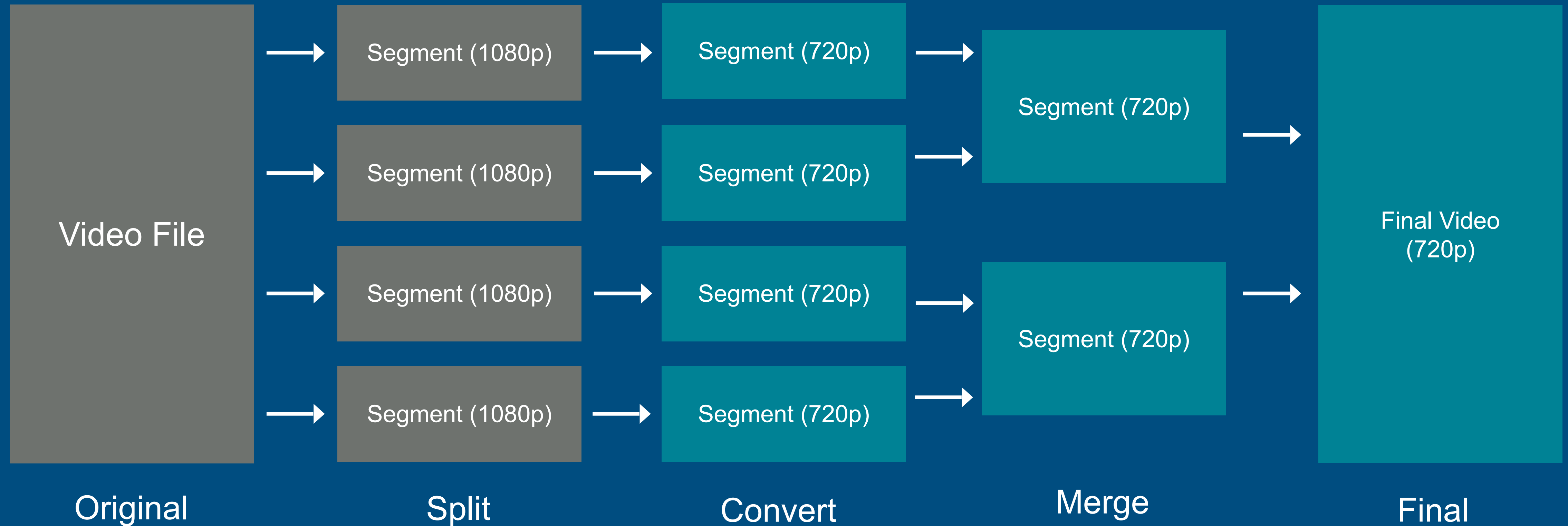
# Parallel Architecture

## Dealing with unexpected problems

- Take a complex problem and solve it with Lambda by applying techniques like MapReduce & Parallelisation
- Can you transcode (i.e. encode) a large video file with a Serverless-only approach?

# Divide and conquer

## Using the Lambda supercomputer

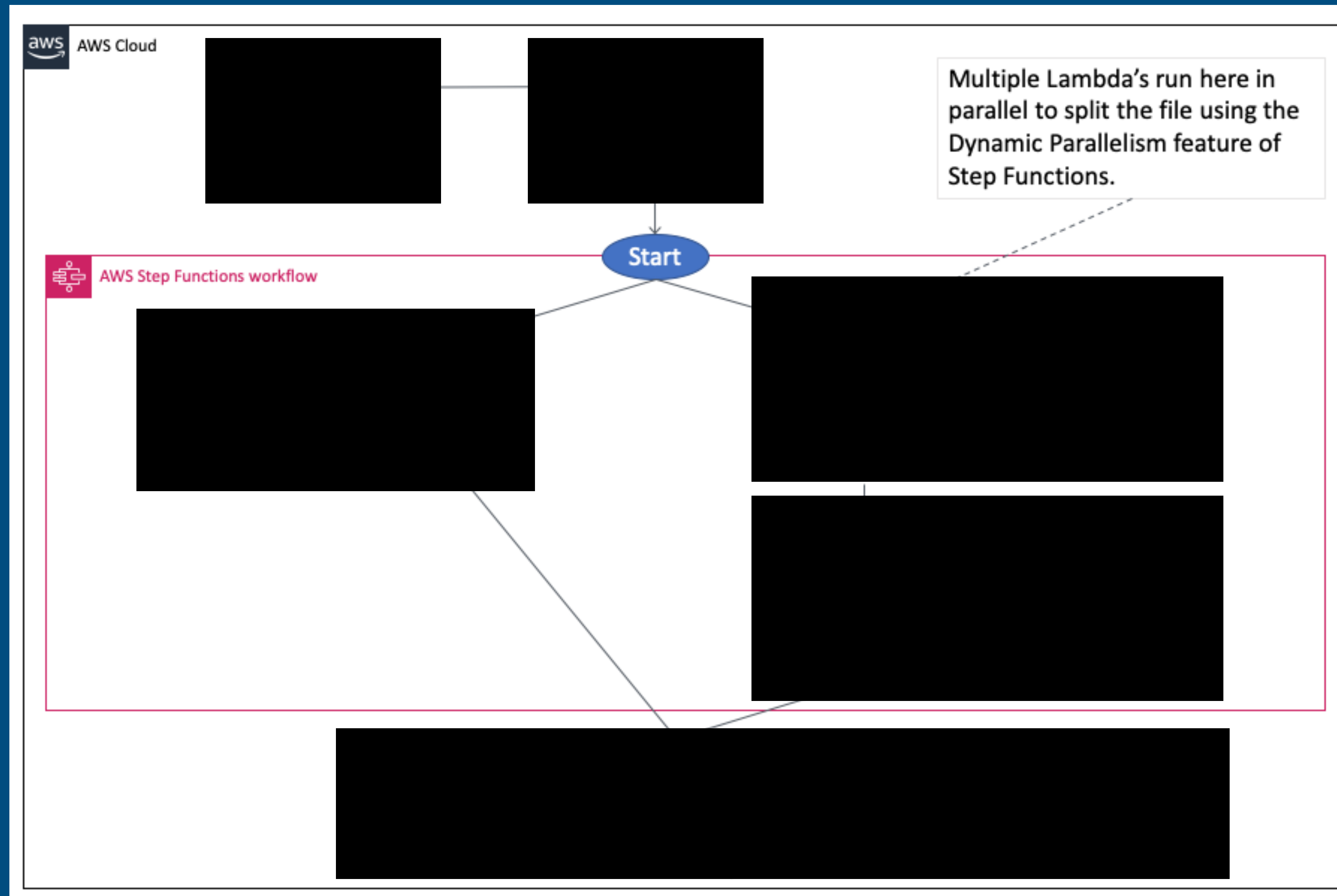


Read more: <https://bit.ly/3wJOdvQ>



# Parallel Computing with Lambda & Step Functions

Read more: <https://bit.ly/3wJOdvQ>



# Serverless Video Transcoder

## Parallel and conquer

	Serverless Lambda	Traditional EC2 (t2.large)	MacBook Pro 16GB 3.5GHz i7
34MB MP4 (00:43, 1920x1080)	<b>11 seconds</b>	32 seconds	18 seconds
77MB MP4 (6:49, 2048x1152)	<b>26 seconds</b>	144 seconds	78 seconds
100MB MP4 (59:56, 1280x720)	<b>86 seconds</b>	1073 seconds	592 seconds
350MB MP4 (07:45, 2560x1440)	<b>35 seconds</b>	432 seconds	224 seconds
420MB MKV (01:02, 3840 x 1606)	112 seconds	157 seconds	<b>101 seconds</b>
1GB MKV (57:57, 1280 x 718)	<b>185 seconds</b>	4320 seconds	2367 seconds

Read more: <https://bit.ly/3wJOdvQ>

# Common Complaints

Why can't things just be easy

- Hard to dev locally ✓
- Hard to debug ✓
- Hard to observe and monitor ✓
- Hard or impossible to do certain things (e.g. long-running tasks) ✓
- Lock-in is a problem ?

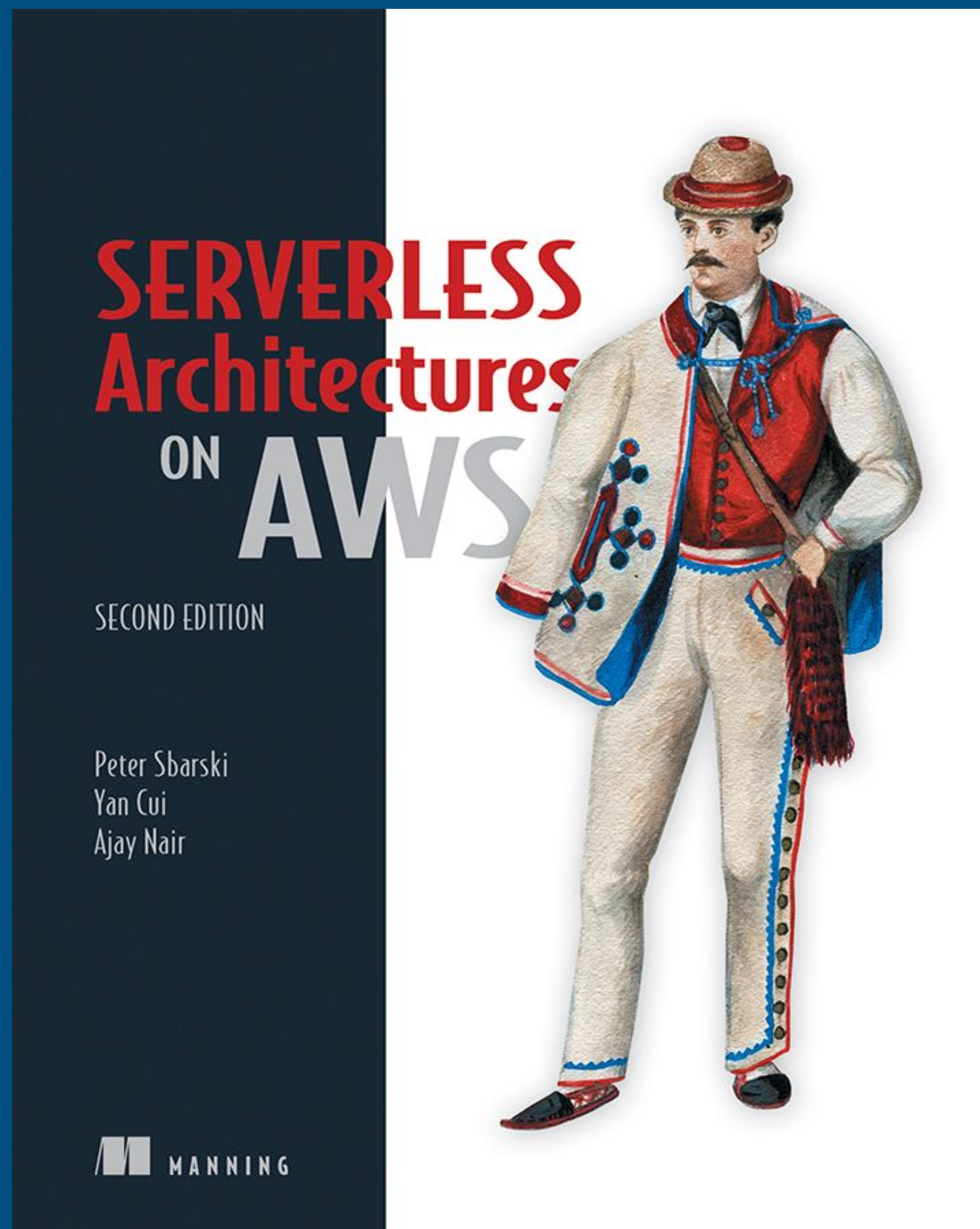


# Modern Applications

## Some lessons were learnt

- Security/compliance first
- Use microservices
- Serverless where possible
- CI/CD
- Monitor, monitor, monitor!
- <https://youtu.be/IPOvrK3S3gQ>
- Serverless monoliths can be OK!
- Automation is a must
- Think through your testing strategy
- Experimentation and architectural changes are easier
- Serverless (& services) > containers

# Thank you



Yan



Ajay

Book: <https://mng.bz/z5mA>

Fatfire: <https://fatfireapp.com>